Data Distribution & Processing CSCI
Data Distribution CSC
Requirements and Design Specification

October 24, 1997
Version 1.1

1. Data Distribution & Processing CSCI

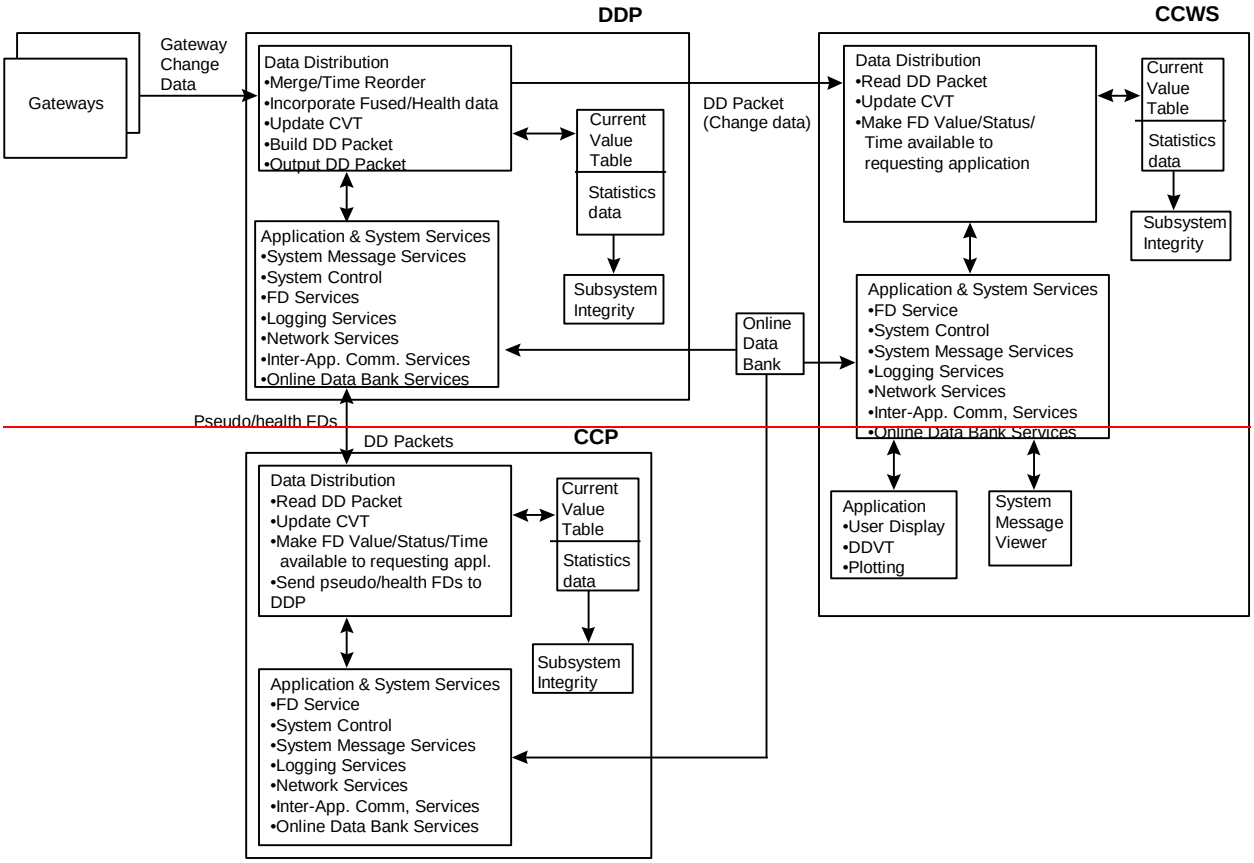
The Data Distribution & Processing CSCI is composed of the following CSCs:
Data Distribution CSC, Data Fusion CSC, and Data Health CSC.

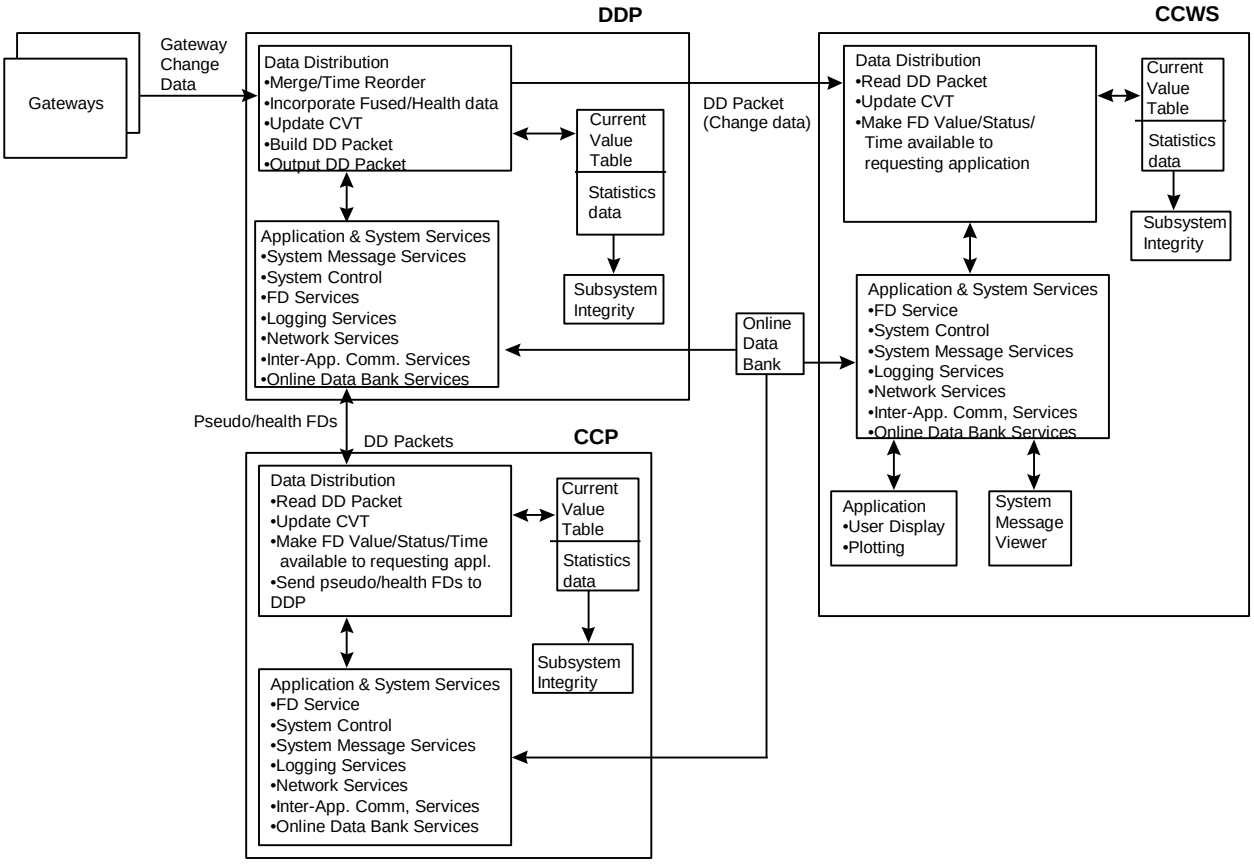
1.1 Data Distribution CSC Introduction

1.1.1 Data Distribution CSC Overview

The Data Distribution CSC resides in the Data Distribution Processor (DDP), the Command and Control Workstations (CCWSs), and the Command and Control Processor (CCPs). The Data Distribution CSC running in the DDP provides the capability to read FD data from the Gateways and distribute it to the RTCN and DCN. The Data Distribution CSC running in the CCPs and *CCWS* provides the capability to receive FD data from DDP and make it available to Command, user applications, and user displays.

Data Distribution CSC Overview is as follows:


DDP
CCWS
Gateways
Gateway Change Data
Data Distribution
•Merge/Time Reorder
•Incorporate Fused/Health data
•Update CVT
•Build DD Packet
•Output DD Packet
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•System Message Services
•System Control
•FD Services
•Logging Services
•Network Services
•Inter-App. Comm. Services
•Online Data Bank Services
DD Packet (Change data)
Online Data Bank
Data Distribution
•Read DD Packet
•Update CVT
•Make FD Value/Status/ Time available to requesting application
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•FD Service
•System Control
•System Message Services
•Logging Services
•Network Services
•Inter-App. Comm, Services
•Online Data Bank Services
Application
•User Display
•DDVT
•Plotting
System Message Viewer
Pseudo/health FDs
DD Packets
CCP
Data Distribution
•Read DD Packet
•Update CVT
•Make FD Value/Status/Time available to requesting appl.
•Send pseudo/health FDs to DDP
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•FD Service
•System Control
•System Message Services
•Logging Services
•Network Services
•Inter-App. Comm, Services
•Online Data Bank Services


DDP
CCWS
Gateways
Gateway Change Data
Data Distribution
•Merge/Time Reorder
•Incorporate Fused/Health data
•Update CVT
•Build DD Packet
•Output DD Packet
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•System Message Services
•System Control
•FD Services
•Logging Services
•Network Services
•Inter-App. Comm. Services
•Online Data Bank Services
DD Packet (Change data)
Online Data Bank
Data Distribution
•Read DD Packet
•Update CVT
•Make FD Value/Status/ Time available to requesting application
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•FD Service
•System Control
•System Message Services
•Logging Services
•Network Services
•Inter-App. Comm, Services
•Online Data Bank Services
Application
•User Display
•Plotting
System Message Viewer
Pseudo/health FDs
DD Packets
CCP
Data Distribution
•Read DD Packet
•Update CVT
•Make FD Value/Status/Time available to requesting appl.
•Send pseudo/health FDs to DDP
Current Value Table
Statistics data
Subsystem Integrity
Application & System Services
•FD Service
•System Control
•System Message Services
•Logging Services
•Network Services
•Inter-App. Comm, Services
•Online Data Bank Services

1.1.2 Data Distribution CSC Operational Description

The Data Distribution CSC supports end-to-end data flow from the point where data is received at the DDP from the Gateway over the RTCN, to the point where data is delivered to the user applications and/or user displays.

1.2 Data Distribution CSC Specifications

The following specifications only cover the capabilities documented in the DD/DH/DF DP1 assessments. New/potential requirements levied on Data Distribution after the DP1 presentation are not addressed here.

1.2.1 Data Distribution CSC Ground Rules

1. For Redstone delivery, Data Distribution will not request a resend of data from its data source, assuming no data will be dropped via the use of Reliable Messages Capability (Resend request to Redundancy Management will be addressed in a later delivery).
2. Each Gateway stream will have an unique multicast stream name.
3. Stream names will be defined as part of activity definition. ~~, for a given activity will be read from a file for Redstone delivery. A long term solution of including that as part of Activity definition will be addressed in a later delivery.~~
4. Data packet received from the same Gateway is already in "time-order."
5. No time validation will be performed on the packets from the Gateways.
6. Each Gateway will send data packets to the RTCN at the System Synchronous Rate (SSR).
7. Time intervals between Gateway sends for the same SSR cycle will be handled by the Gateways. Data Distribution processing will be completely data driven. DD output rate is based on input rate coming from the Gateways. Data will be output to RTCN as soon as DD processing is complete.
8. An empty packet will be sent by the Gateway if there is no data changed within an SSR cycle.
9. ~~*Display Attributes processing for Redstone is not supported.*~~
10. All applications, with exception of ~~*Data Fusion,*~~ Data Health, and Constraint Manager, will interface with Data Distribution via FD Services.
12. Data Distribution will access the OLDB via FD Services.
13. ~~*Application derived FDs will not be supported for Redstone.*~~
14. ~~*Time Homogeneous Data Sets (THDS) will not be supported for Redstone.*~~
15. ~~*The FD value override capability will not be provided for Redstone.*~~
16. *Persistent data/checkpoint will not be supported (future requirement).*
17. ~~*Interface with Constraint Management will not be supported for Redstone.*~~
18. ~~*Data refresh will not be supported for Redstone.*~~
19. *Redundancy management will not be supported (future requirement).*
20. The Gateway will send the THDS data in the format documented in the Thor RTPS Packet Payload ICD. ~~.~~
21. Each THDS will have a THDS set FD.
22. When the request is made with FDID for the THDS set FD, the associated THDS set is returned to the requesting application.

23. FD data value, reason code value, and status bit from Gateway will be stored in CVT as received. No checking will be performed by Data Distribution.

1.2.2 Data Distribution CSC Functional Requirements

1.2.2.1 Data Distribution (DD) at the DDP

1. DD will receive Gateway Changed Data Packets on the RTCN.
2. DD will support at least 20 Gateways simultaneously on a single DDP.
3. DD will provide the capability to perform time-reordering on the data packets received across all the Gateways.
4. DD will provide the capability to perform time-reordering based on the time value(s) provided in the DD Packet Payload data.
5. DD will send a system message if no packet is received from a Gateway at the end of the System Synchronous Rate (SSR) cycle.
6. DD will merge data packets received from the RTCN.
7. DD will provide the capability to build Data Distribution Packets based on the CLCS DD Packet Payload format.
8. DD will distribute the following information in the Data Distribution packets:
   a. Data value
   b. Health Data status
   c. Timestamp
   d. Display Attribute
9. DD will provide the capability to output Data Distribution packets to the RTCN at the System Synchronous Rate (SSR).
10. DD will provide the capability to buffer Data Distribution packets and output DD packets to the DCN at the Display Synchronous Rate (DSR).
11. DD will provide the capability to receive Pseudo FD data packets from all CCPs.~~.~~
12. DD will provide the capability to merge Gateway Change data and Pseudo FD change data into a single stream.
13. DD will not update the CVT for fused FD if~~id~~ the fusion process for that FD is inhibited.
14. DD will provide the capability to mark all associated FDs to "fail" when the Gateway becomes inactive.

1.2.2.2 Data Distribution at the CCP

1. DD will receive Data Distribution packets on the RTCN.
2. DD will send a system message if no packet is received from the RTCN at the end of the SSR.
3. DD will provide the capability to collect Pseudo FDs.~~.~~
4. DD will provide the capability to~~o~~ build Change Data packets for Pseudo FDs based on the CLCS DD packet payload format

5. DD will provide the capability to output Pseudo FD Change Data packets to DDP at System Synchronous Rate (SSR).

1.2.2.3 Data Distribution at the CCWS

1. DD will receive Data Distribution packets on the DCN.
2. DD will send a system message if no packet is received from the DCN at the end of a DSR cycle.

1.2.2.4 Data Distribution (DD) at the DDP/CCP/CCWS

1. DD will initialize the CVT with the information contained in the Online Data Bank (OLDB).
2. DD will provide the capability to update the CVT with changed data.
3. DD will provide the capability to update the CVT with FD data:
   a. Raw or EU data
   b. Health Data Values
   c. Status/State Values
   d. Timestamp
   e. Display Attribute
4. DD will provide an application interface allowing an application to:
   a. subscribe to FD data
   b. publish FD data
5. DD will provide an application interface allowing applications to access, on demand, the following data for an FD or a list of FDs:
   a. Raw or EU data
   b. Health Data Values
   c. Status/State Values
   d. Timestamp
   e. Display Attribute
6. DD will provide an application interface allowing applications to access, on a "change only" basis, the following data for a FD or a list of FDs:
   a. Raw or EU data
   b. Health Data Values
   c. Status/State Values
   d. Timestamp
   e. Display Attribute
7. DD will provide the capability to maintain statistics on packet rates and data rates in an internal table.
8. DD will support processing of data in a Time Homogeneous Data Set (THDS).

9. DD will provide an application interface allowing applications to assign the following to a FD or a list of FDs:
    a. Data Value.
    b. Display Attribute
10. DD will notify Subsystem Integrity immediately whenever a failure occurs that causes Data Distribution to cease processing change data.

11. DD will provide an application program interface for Subsystem Integrity to obtain the following information:
    a. Overall health of Data Distribution.
    b. Error situations (e.g., Failure to setup network communication).
    c. Statistics on packet size and data rate.
12. DD will provide an application program interface allowing applications to activate/inhibit fusion FD processing.
13. DD will provide a mechanism to notify Constraint Manager when data is ready for processing.

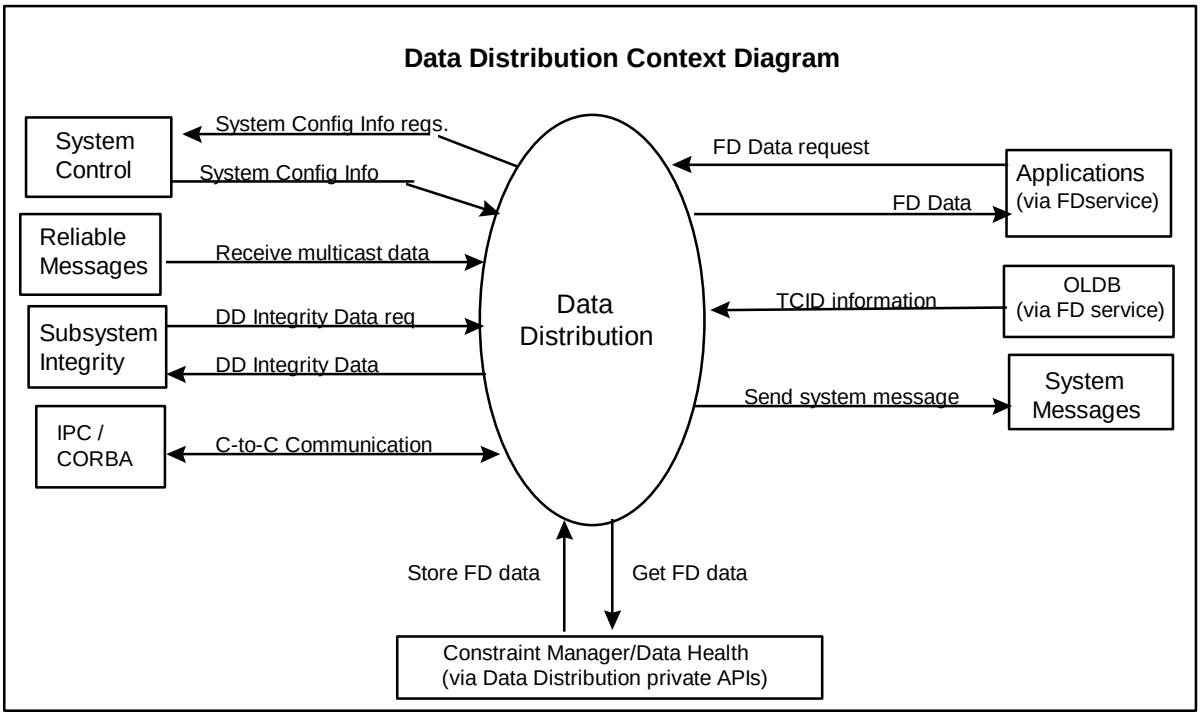
1.2.2.5 Data Distribution (DD) Test Tools

1. DD will provide the capability to construct and feed data streams to Data Distribution residing on the same platform.
2. DD will provide an Graphic User Interface (GUI) allowing user to:
    a. Subscribe to a list of FDs and display values on the screen.
    b. Publish a list of FDs at specific rate.
    c. Retrieve data packets from the DCN or RTCN and print the header time, length of packet and number of packets being sent.
3. DD will provide the capability to display the CVT contents.

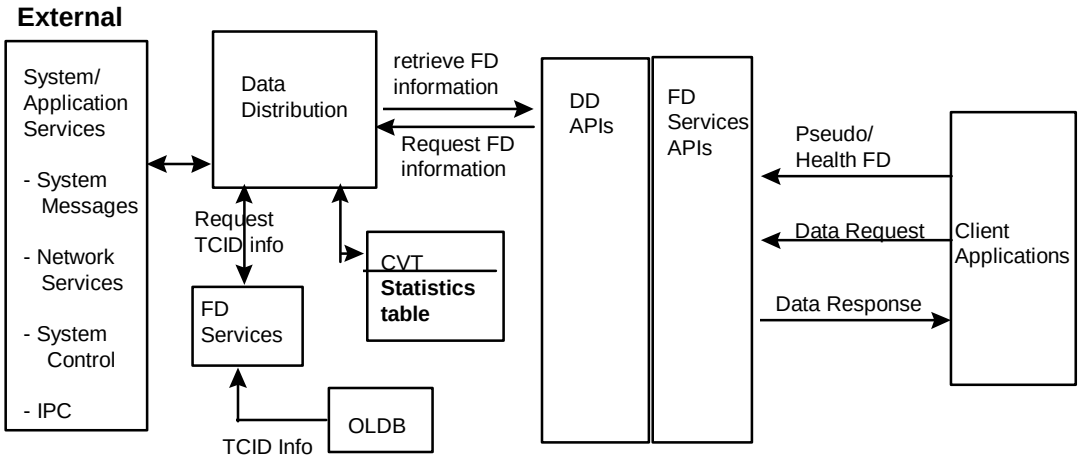
1.2.3 Data Distribution CSC Performance Requirements

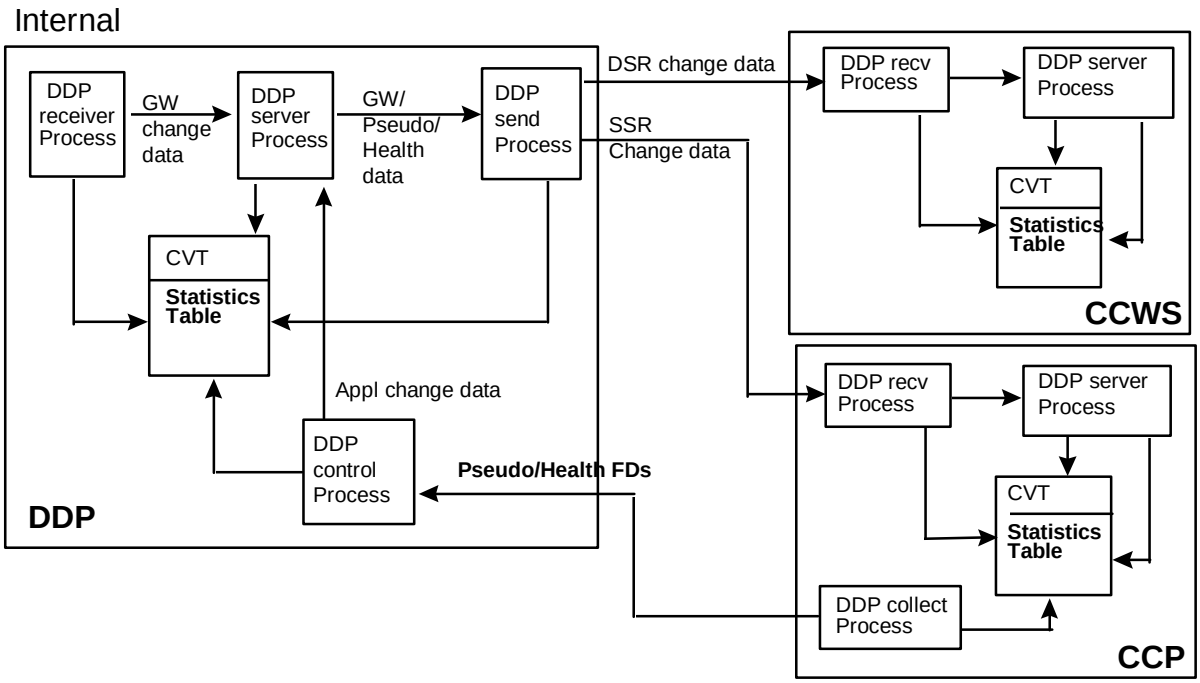
1. DD will be able to process up to 25,000 change FDs from the Gateway(s) per second.
2. DD will be able to process up to 50,000 change FDs from the Gateway(s) in a given second without losing any data.
3. DD will be able to incorporate 5,000 processed FDs (fused FDs and Health status) into the CVT per second.

1.2.4 Data Distribution CSC Interfaces



1.2.5 Data Distribution CSC Data Flow Diagram





1.3 Data Distribution CSC Design Specification

The Data Distribution CSC supports end-to-end data flow from the point where data is received at the DDP from the Gateways over the RTCN, to the point where data is delivered to the user applications and/or user displays.

The Data Distribution CSC on the DDP will perform these functions:

1. Receive gateway changed data packets on the RTCN
2. Perform time-reordering on the data packets received from the Gateways
3. Buffer distributed data packets and output packets to the DCN at the DSR
4. Buffer distributed data packets and output packets to the RTCN at the SSR
5. Provide data to the Data Fusion CSC and the Data Health CSC
6. Incorporate fusion and health data into the data stream
7. Receive pseudo~~fusion~~/health FDs~~/display attributes~~ from the CCPs.

The Data Distribution CSC on the CCWS will perform these functions:

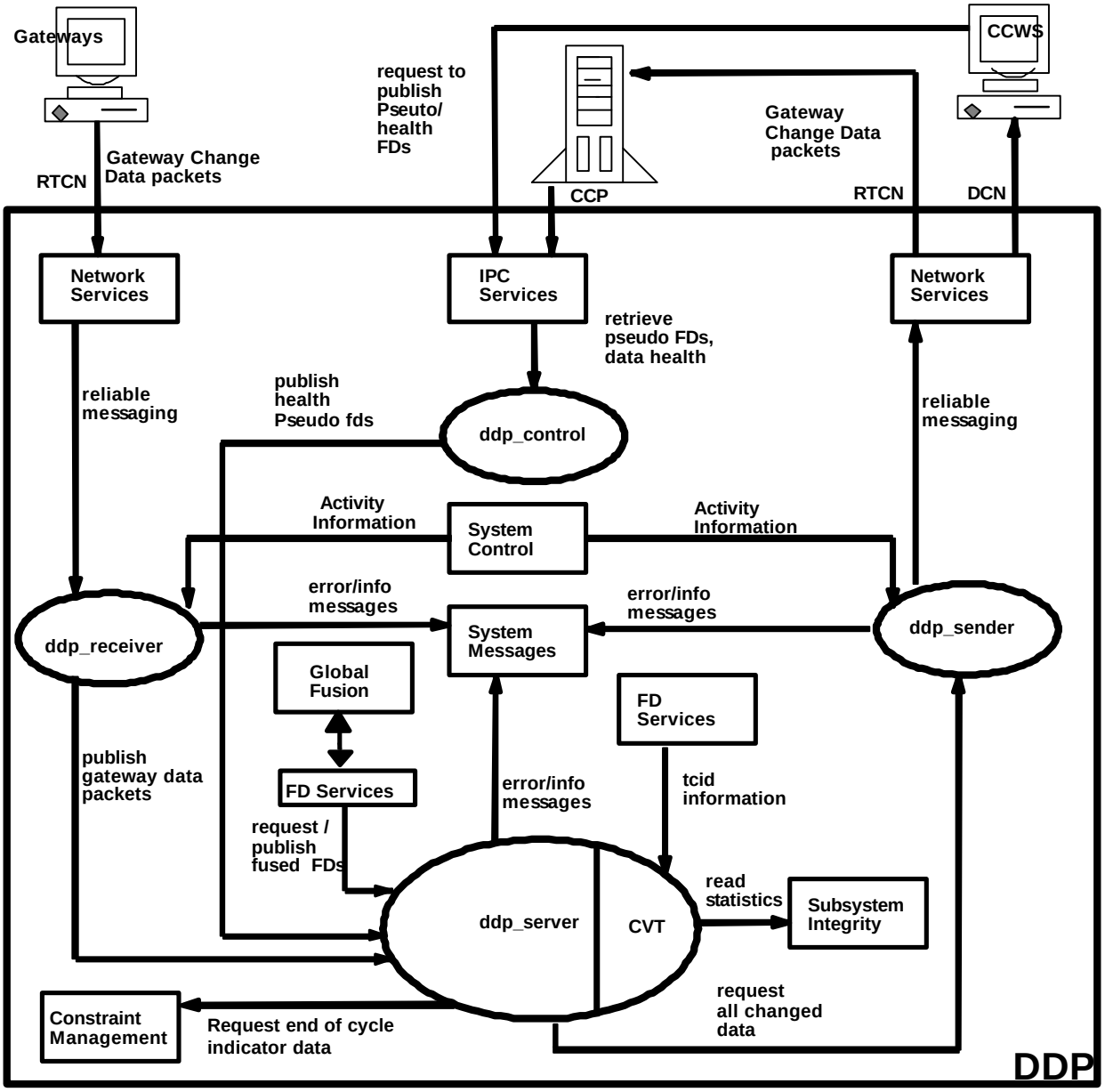
1. Receive distributed data packets on the DCN
2. Distribute packets to clients (i.e. display applications, viewers, user applications)

The Data Distribution CSC on the CCP will perform these functions

1. Receive distributed data packets on the RTCN
2. Distribute packets to clients (i.e. end item managers)
3. Publish pseudo~~display attributes~~/health FDs~~/fusion~~ to the DDP at SSR.

1.3.1 Data Distribution Detailed Data Flow

1.3.1.1 DDP Detailed Data Flow



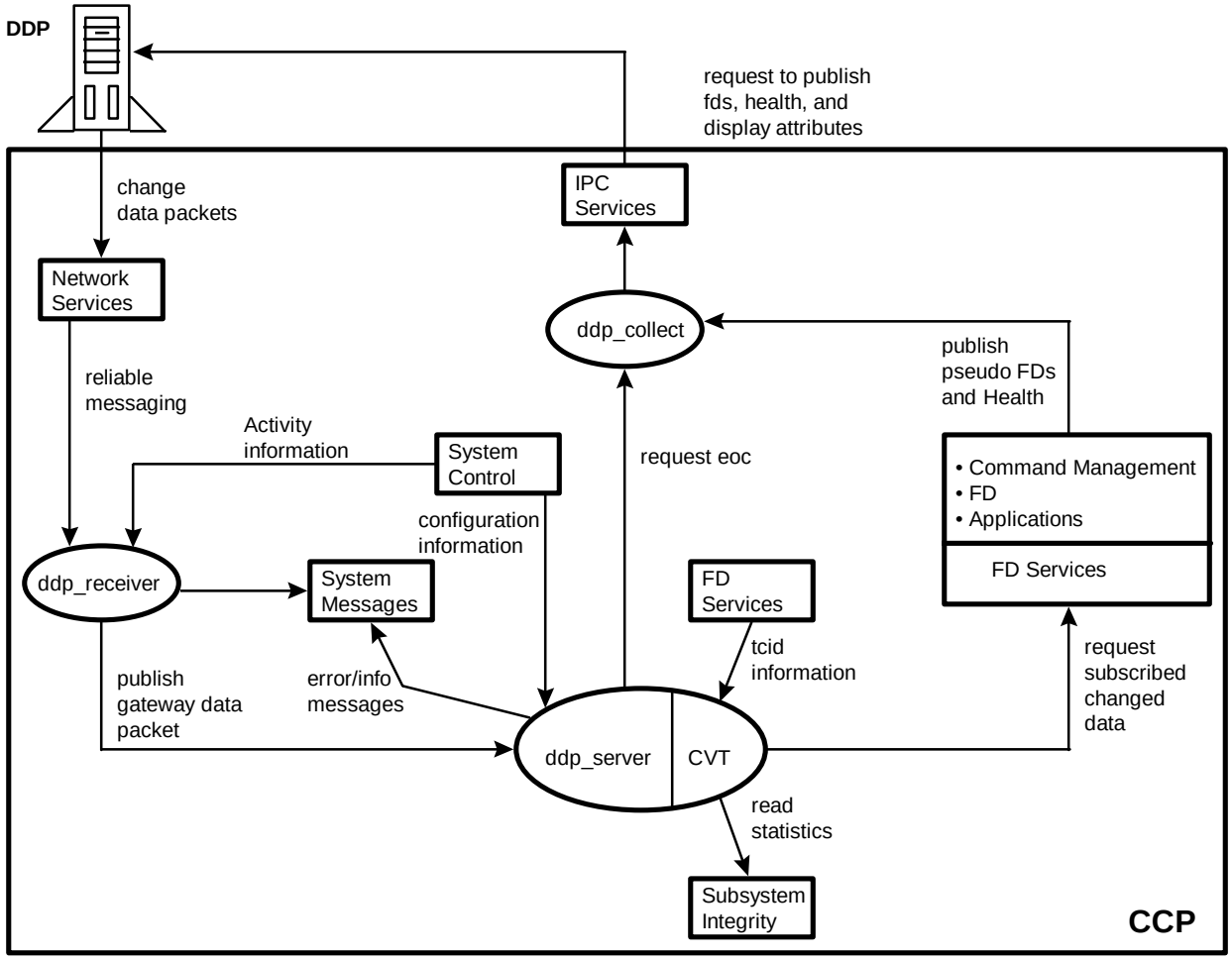
Data Distribution processes are started by System Control Unattended Server Remote Configuration (USRC) at platform initialization via the Platform Startup Script List (PSSL). Data Distribution processes will be terminated by System Control upon platform termination.

The Data Distribution CSC interfaces with Network Services both to receive data from the Gateways, and send data out to the CCP and CCWSs. Error conditions that are encountered by Data Distribution are written to System Messaging. At initialization, System Control provides the configuration information to build the multicast addresses, and determine the number of Gateways. Subsystem Integrity interfaces with data distribution to retrieve information in the CVT header containing statistical information on the health of the system.

Data is merged from the Gateways in the ddp_receiver process and sent in time order to the ddp_server. The ddp_server process stores data in the current value table and sends all changed values to the ddp_sender process and the global fusion processes. The ddp_sender process buffers the changed values into SSR buffers and DSR buffers to output to the CCP and CCWSs respectively. The ddp_control process receives remote requests and publishes data health and, pseudo~~fused~~ FDs~~, and display attributes~~ to the CVT.

When a gateway becomes inactive, the health bit for the associated FDs is set to "fail" and the reason code is set to "02", indicating the gateway is down.

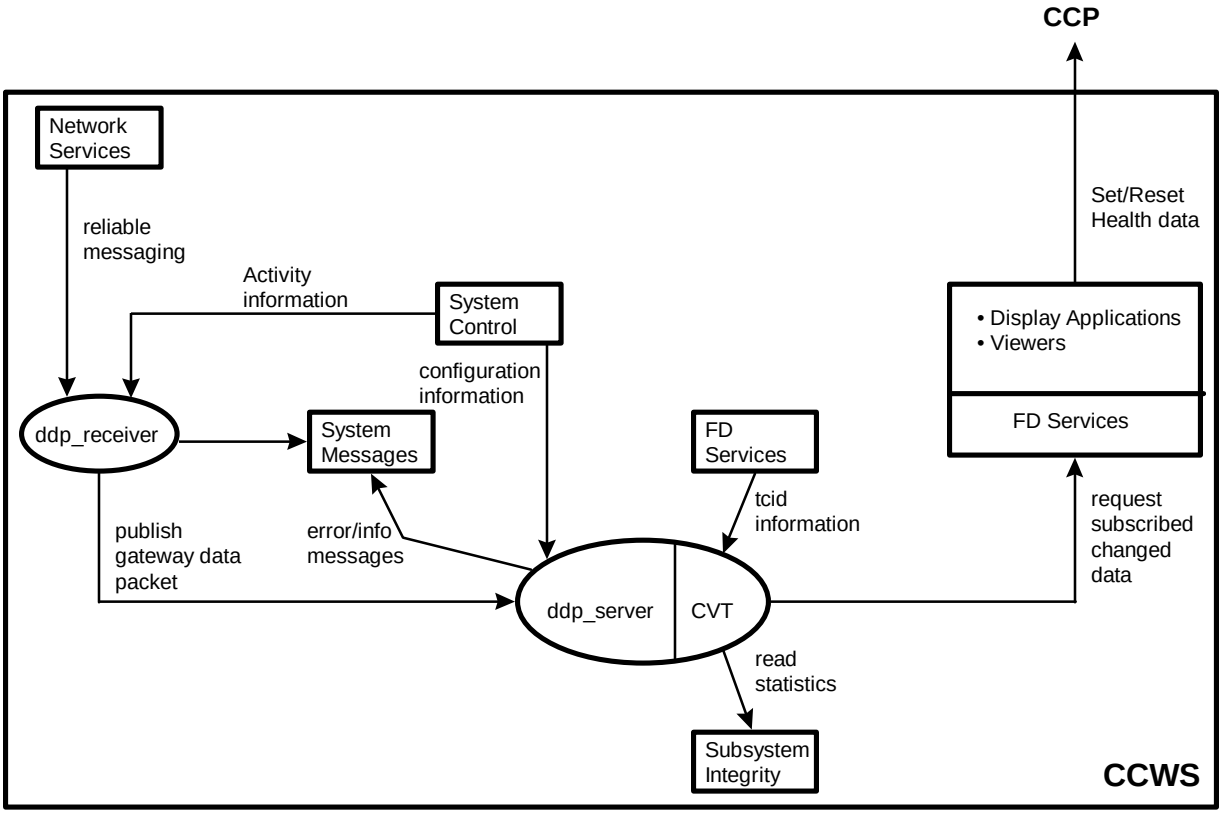
1.3.1.2 CCP Detailed Data Flow



Data Distribution processes are started by System Control Unattended Server Remote Configuration (USRC) at platform initialization via the Platform Startup Script List (PSSL). Data Distribution processes will be terminated by System Control upon platform termination.

The ddp_receiver process on the CCP receives data from the DDP at 10ms (RTCN SSR) rate on the CCP. The ddp_receiver parses the data buffer and publishes the changed data values to the ddp_server. The ddp_server process stores the changed values in the CVT. The ddp_server sends the changed values to the clients (i.e display applications, viewers, and end item managers). The ddp_collect process receives requests from applications and publishes the requests to the DDP at the SSR.

1.3.1.3 CCWS Detailed Data Flow



Data Distribution processes are started by System Control Unattended Server Remote Configuration (USRC) at platform initialization via the Platform Startup Script List (PSSL). Data Distribution processes will be terminated by System Control upon platform termination.

The ddp_receiver process on the CCWS receives data from the DDP at 100ms (DCN DSR) rate. The ddp_receiver parses the data buffer and publishes the changed data values to the ddp_server. The ddp_server process stores the changed values in the CVT. The ddp_server sends the changed values to the clients (i.e display applications, viewers, and end item managers).

1.3.2 Data Distribution External Interfaces

1.3.2.1 Data Distribution Message Formats

1.3.2.1.1 ddp_receiver messages

1.
Message Group = DDP
Severity = Informational

DDP Receiver is initialized

Help Information Content:
The receiver process has initialized successfully

Detailed Information:
n/a

2.
Message Group = DDP
Severity = Error

DDP Receiver was unable to establish a connection with the server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
During initialization of the DDP services, there was a problem creating a socket connection between the ddp_server and the ddp_receiver.

Detailed Information:
n/a

3.
Message Group = DDP
Severity = Warning

DDP Receiver was unable to send an event to the ddp_server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_receiver was unable to send the data packet event to the ddp_server, so there must be a problem with the socket connection

Detailed Information:
Make sure the ddp_server process is still active
Check to see if the queue is full

4.
Message Group = DDP
Severity = Error

DDP Receiver was unable to retrieve the configuration information, system_error = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing System Control error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_receiver was unable to retrieve the configuration information needed to define the multicast address and determine the number of Gateways to be read.

Detailed Information:
See System Control for error conditions

5.
Message Group = DDP
Severity = Error

DDP Receiver was unable to establish a connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_receiver was unable to open the lan connection via ”clm_open” from Network Services

Detailed Information:
Refer to error conditions provided by Network Services

6.
Message Group = DDP
Severity = Informational

DDP Receiver established connection with the LAN, mcaddr = #ARGUMENT1#

ARGUMENT1 = multicast address

Help Information Content:
n/a

Detailed Information:
n/a

7.
Message Group = DDP
Severity = Warning

DDP Receiver lost the connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_receiver was unable to receive data via ”clm_recv” from Network Services.

Detailed Information:
Refer to error conditions provided by Network Services

8.
Message Group = DDP
Severity = Informational

DDP Receiver detected that gateway #ARGUMENT1# has become inactive

ARGUMENT1 = Multicast address for gateway that went inactive

Help Information Content:
The ddp_receiver was unable to receive data from a gateway after TBD milliseconds

Detailed Information:
Check to make sure the gateway is sending out data

9.
Message Group = DDP
Severity = Informational

DDP Receiver detected that gateway #ARGUMENT1# has become active

ARGUMENT1 = Multicast address for gateway that became active

Help Information Content:
The ddp_receiver is received data after it was previously inactive

Detailed Information:
n/a

10.
Message Group = DDP
Severity = Informational

DDP Receiver is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.2 ddp_server messages

11.
Message Group = DDP
Severity = Informational

DDP Server is initialized

Help Information Content:
The server process has initialized successfully

Detailed Information:
n/a

12.
Message Group = DDP
Severity = Error

DDP Server detected an error reading from the OLDB, oldb_error = #ARGUMENT1#, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing the FD services error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_server was unable to read the online data bank information for initializing the CVT

Detailed Information:
See fd_services error numbers for information

13.
Message Group = DDP
Severity = Error

DDP Server detected an error creating the CVT, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_server was unable to allocate shared memory to create the current value table.

Detailed Information:
Use the UNIX errno to determine the problem

14.
Message Group = DDP
Severity = Error

DDP Server lost the connection with a client, name = #ARGUMENT1#, pid = #ARGUMENT2#

ARGUMENT1 = Process name of client
ARGUMENT2 = Process id of client

Help Information Content:
The ddp_server lost the socket connection with the client. The client will detect it and try to reconnect.

Detailed Information:
n/a

15.
Message Group = DDP
Severity = Informational

DDP Server is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.3 ddp_sender messages

16.
Message Group = DDP
Severity = Informational

DDP Sender is initialized

Help Information Content:
The sender process has initialized successfully

Detailed Information:
n/a

17.
Message Group = DDP
Severity = Error

DDP Sender was unable to establish a connection with the server, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
During initialization of the DDP services, there was a problem creating a socket connection between the ddp_server and the ddp_sender.

Detailed Information:
n/a

18.
Message Group = DDP
Severity = Error

DDP Sender was unable to retrieve the configuration information, system_error = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing System Control error number value
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_sender was unable to retrieve the configuration information needed to define the multicast address and determine the number of Gateways to be read.

Detailed Information:
See System Control for error conditions

19.
Message Group = DDP
Severity = Informational

DDP Sender established connection with the LAN, mcaddr = #ARGUMENT1#

ARGUMENT1 = multicast address

Help Information Content:
n/a

Detailed Information:
n/a

20.
Message Group = DDP
Severity = Error

DDP Sender lost the connection with the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT2#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing Network Services error number value

Help Information Content:
The ddp_sender was unable to open the lan connection via ”clm_open” from Network Services.

Detailed Information:
Refer to error conditions provided by Network Services

21.
Message Group = DDP
Severity = Warning

DDP Sender was unable to write to the LAN, mcaddr = #ARGUMENT1#, errno = #ARGUMENT1#

ARGUMENT1 = multicast address
ARGUMENT2 = unsigned integer representing UNIX error number value

Help Information Content:
The ddp_sender was unable to write to the multicast address.

Detailed Information:
Make sure the multicast address is still open.

22.
Message Group = DDP
Severity = Informational

DDP Sender is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.4 ddp_control messages

23.
Message Group = DDP
Severity = Informational

DDP Control is initialized

Help Information Content:
The control process has initialized successfully

Detailed Information:
n/a

24.
Message Group = DDP
Severity = Error

DDP Control was unable to establish a connection with IPC services, errno = #ARGUMENT1#, es_errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing UNIX error number value
ARGUMENT2 = unsigned integer representing IPC error number value

Help Information Content:
During initialization, there was a problem establishing a connection with IPC services. Refer to IPC for the failure indicated.

Detailed Information:
n/a

25.
Message Group = DDP

Severity = Warning

DDP Control was unable to read from IPC Servies, errno = #ARGUMENT1#, es_errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing UNIX error number value
ARGUMENT2 = unsigned integer representing IPC services error number value

Help Information Content:
The ddp_control was unable to read from IPC services

Detailed Information:

26.
Message Group = DDP
Severity = Error

DDP Control was unable to open the data health file, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
n/a

Detailed Information:
n/a

27.
Message Group = DDP
Severity = Warning.

DDP Control was unable to process the data health file, errno = #ARGUMENT1#

ARGUMENT1 = unsigned integer representing UNIX error number value

Help Information Content:
n/a

Detailed Information:
n/a

28.
Message Group = DDP
Severity = Informational

DDP Control received an invalid message, action = #ARGUMENT#

Help Information Content:
n/a

Detailed Information:
n/a

29.
Message Group = DDP
Severity = Informational

DDP Control is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.1.5 ddp_collect messages

30.
Message Group = DDP
Severity = Informational

DDP Collect is initialized

Help Information Content:
The collect process has initialized successfully

Detailed Information:
n/a

31.
Message Group = DDP
Severity = Error

DDP Collect was unable to establish a connection with IPC services, errno = #ARGUMENT1#, es_errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing UNIX error number value
ARGUMENT2 = unsigned integer representing IPC error number value

Help Information Content:
During initialization, there was a problem establishing a connection with IPC services. Refer to IPC for the failure indicated.

Detailed Information:
n/a

32.
Message Group = DDP
Severity = Warning

DDP Collect was unable to send to IPC Services, errno = #ARGUMENT1#, es_errno = #ARGUMENT2#

ARGUMENT1 = unsigned integer representing UNIX error number value
ARGUMENT2 = unsigned integer representing IPC services error number value

Help Information Content:
The ddp_collect was unable to read from IPC services

Detailed Information:

33.
Message Group = DDP
Severity = Informational

DDP Collect is terminating

Help Information Content:
n/a

Detailed Information:
n/a

1.3.2.2 Data Distribution Display Formats

There are no display formats for the DDP CSC.

1.3.2.3 Data Distribution Input Formats

There are no input formats for the DDP CSC.

1.3.2.4 Data Distribution Recorded Data

Statistical data will be kept in shared memory and will be accessible via DD API.~~.~~

1.3.2.5 Data Distribution Printer Formats

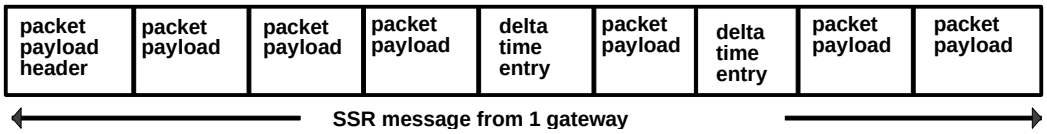
There are no printer formats for the DDP CSC.

1.3.2.6 Data Distribution Inter-process Communication

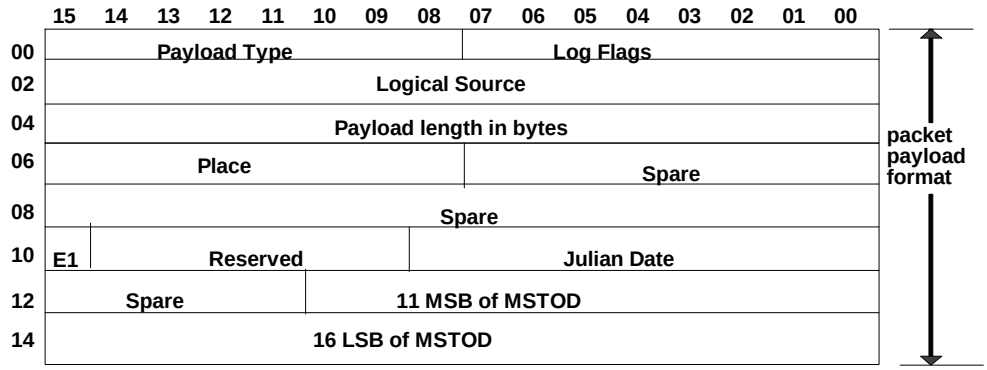
1.3.2.6.1 Network Services - Packet Payload Formats

These are the packet formats received from the Gateway machines as defined in the THOR RTPS Packet Payload ICD, 84K00357-001, Revision:Basic, dated July , 1997.

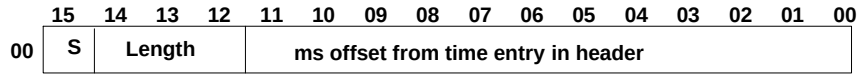
Packet Payload Layout



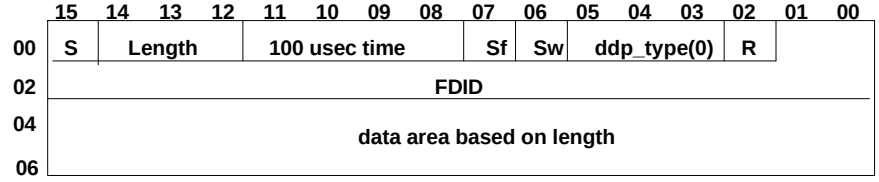
Packet Payload Header



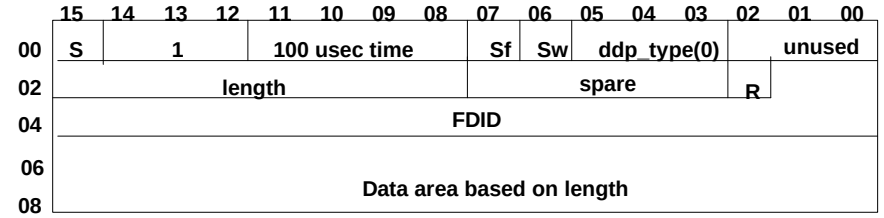
Delta Time Entry



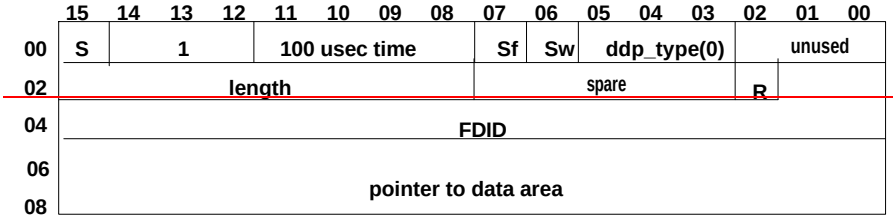
Packet Payload Entry



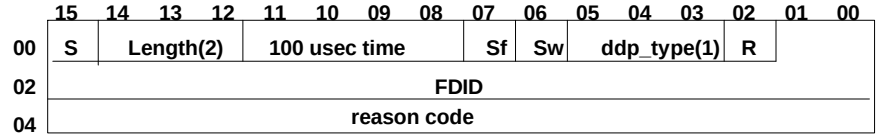
Packet Payload Entry (variable length)



Packet Payload Entry (variable length)



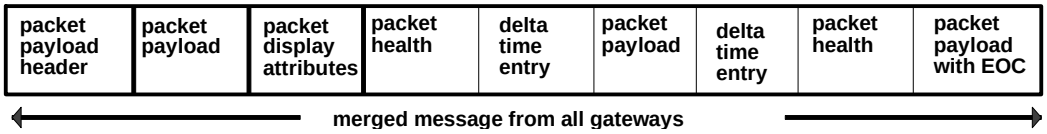
Packet Health Entry



1.3.2.6.2 Distributed Data Packet Messages

This is the packet format that is distributed to the applications for data requests from the ddp_server process.

Distributed Packet Layout



Packet Payload Header

<see Packet Payload Header in section 1.3.2.6.1>

Delta Time Entry

<see Payload Time Entry in section 1.3.2.6.1>

Packet Payload Entry

<see Packet Payload Entry in section 1.2.3.6.1>

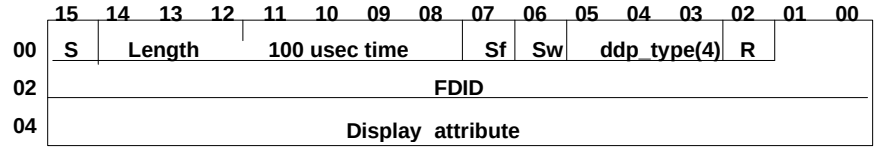
Packet Payload Entry (Variable Lenght)

<see Packet Payload Entry (variable length) in section 1.2.3.6.1>

Packet Payload Health Entry

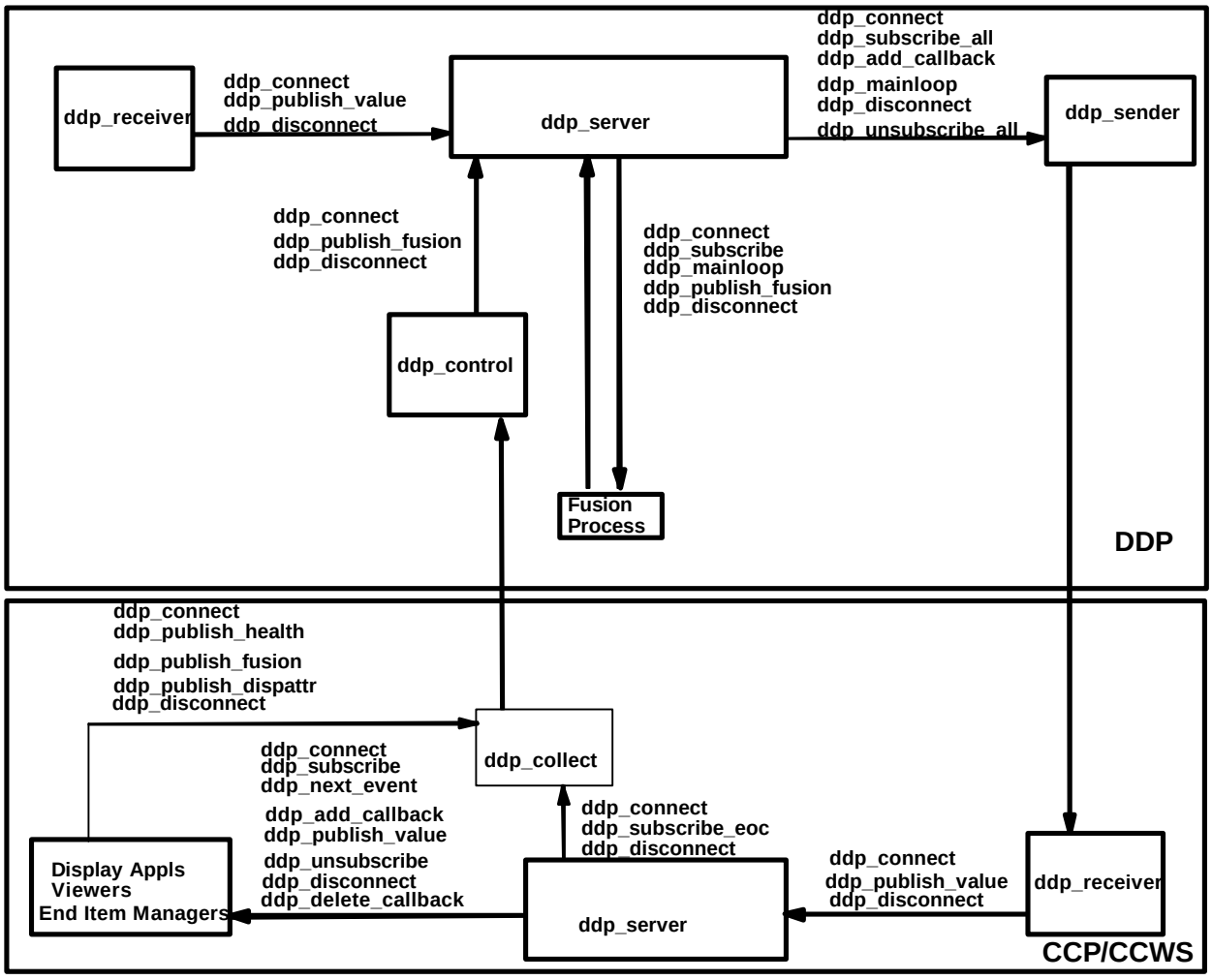
<see Packet Payload Health Entry in section 1.2.3.6.1>

Packet DisplayAttributeEntry



1.3.2.7 Data Distribution External Interface Calls

1.3.2.7.1 Current Value Table Interfaces



Initialization/startup:

1. **ddp_app_add_input**

   Add an event callback function to be invoked when an event arrives in the Xtmainloop
2. **ddp_add_callback**

   Add an event callback function to be invoked when changes arrive
3. **ddp_connect**

   Establishes a client connection with the ddp_server
4. **ddp_get_client_socket**

   Retrieves the client socket identifier
5. **ddp_read_socket**

   Retrieves information from the client socket
6. **ddp_reconnect**

Performs a reconnection to the server if the connection was broken.

7. **ddp_signals**

   Sets a default set of signal handlers

8. **ddp_subscribe**

   Tells the server to notify client when specified FD health/value changes

9. **ddp_subscribe_eoc**

   End of cycle indicator. Tells the server to notify client when data is ready for processing

10. **ddp_subscribe_value**

    Tells the server to notify client when any FD health/value changes. Return the packet in the cvt_value format.

11. **ddp_subscribe_health**

    Tells the server to notify client when any FD health/value changes. Return the packet in the cvt_health format.

12. **ddp_subscribe_record**

    Tells the server to notify client when any FD health/value changes. Return the packet in the cvt_record format.

13. **ddp_subscribe_all**

    Tells the server to notify client when any FD health/value changes. Return the packet in the cvt_value format.

14. **ddp_flush**

    Flush the clients event queue

Operational:

15. **ddp_app_remove_input**

    Delete the callback added from the ddp_app_add_input

16. **ddp_dispatch_event**

    Reads the socket and invokes the callbacks

17. **ddp_mainloop**

    Indefinitely waits for incoming events and invokes callbacks

18. **ddp_next_event**

    Performs a ”select” waiting for incoming events

19. **ddp_publish_dispattr**

    Notifies the ddp_server to store the display attributes~~ are stored~~ in the CVT for an FD

20. **ddp_publish_fusion**

    Notifies the ddp_server to store the fusion result in the CVT for an FD

21. **ddp_publish_health**

    Notifies the ddp_server to store the health reason codes in the CVT for an FD

22. **ddp_inhibit_fd**

    Notifies the ddp_server to inhibit/enable an fused FD

Termination:

23. **ddp_delete_callback**

    Removes the callback from the event notification

24. **ddp_disable_fds**

    Notifies the server to stop sending events for changed FDs

25. **ddp_disconnect**

    Performs disconnection from the server and close the socket connection

26. **ddp_unsubscribe**

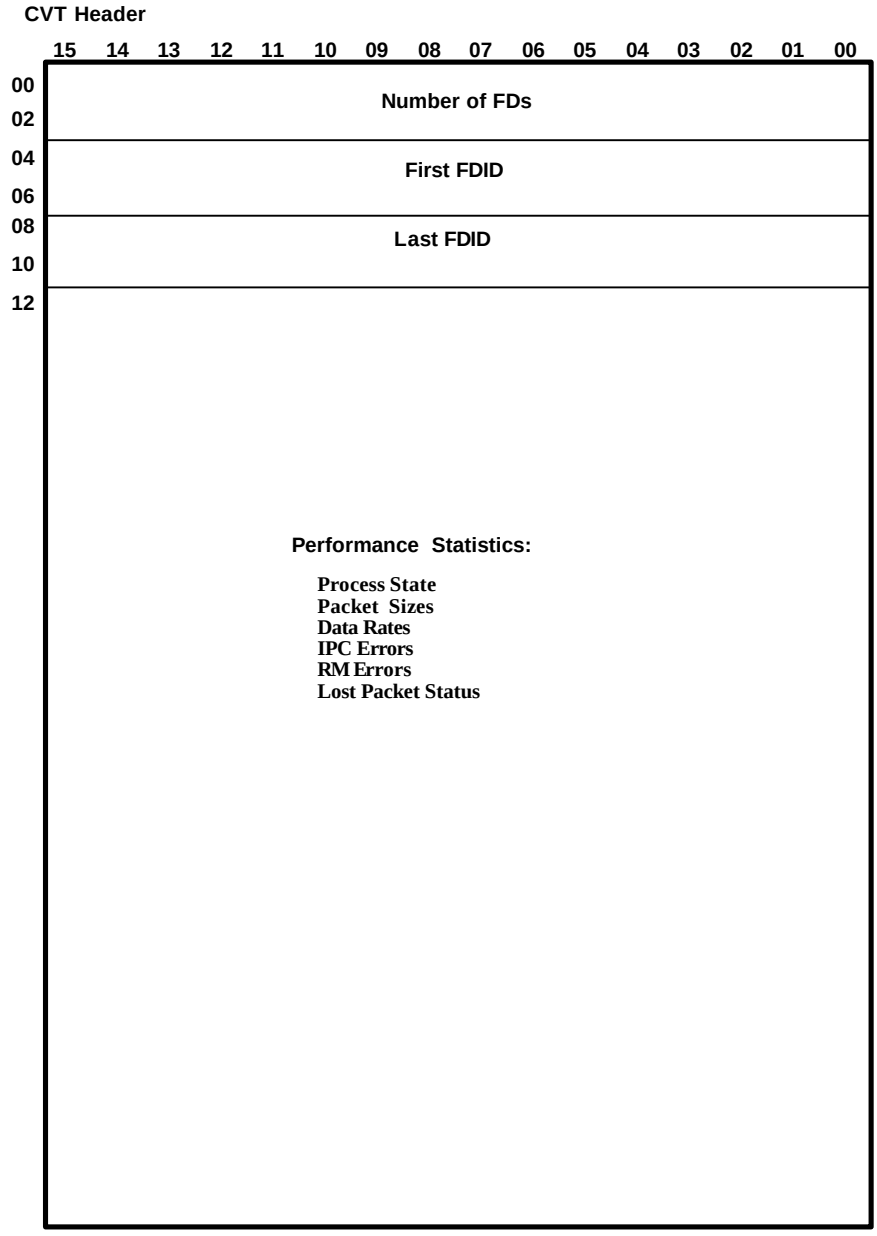
    Notifies the ddp_server to unsubscribe to specified FDs

27. **ddp_unsubscribe_all**

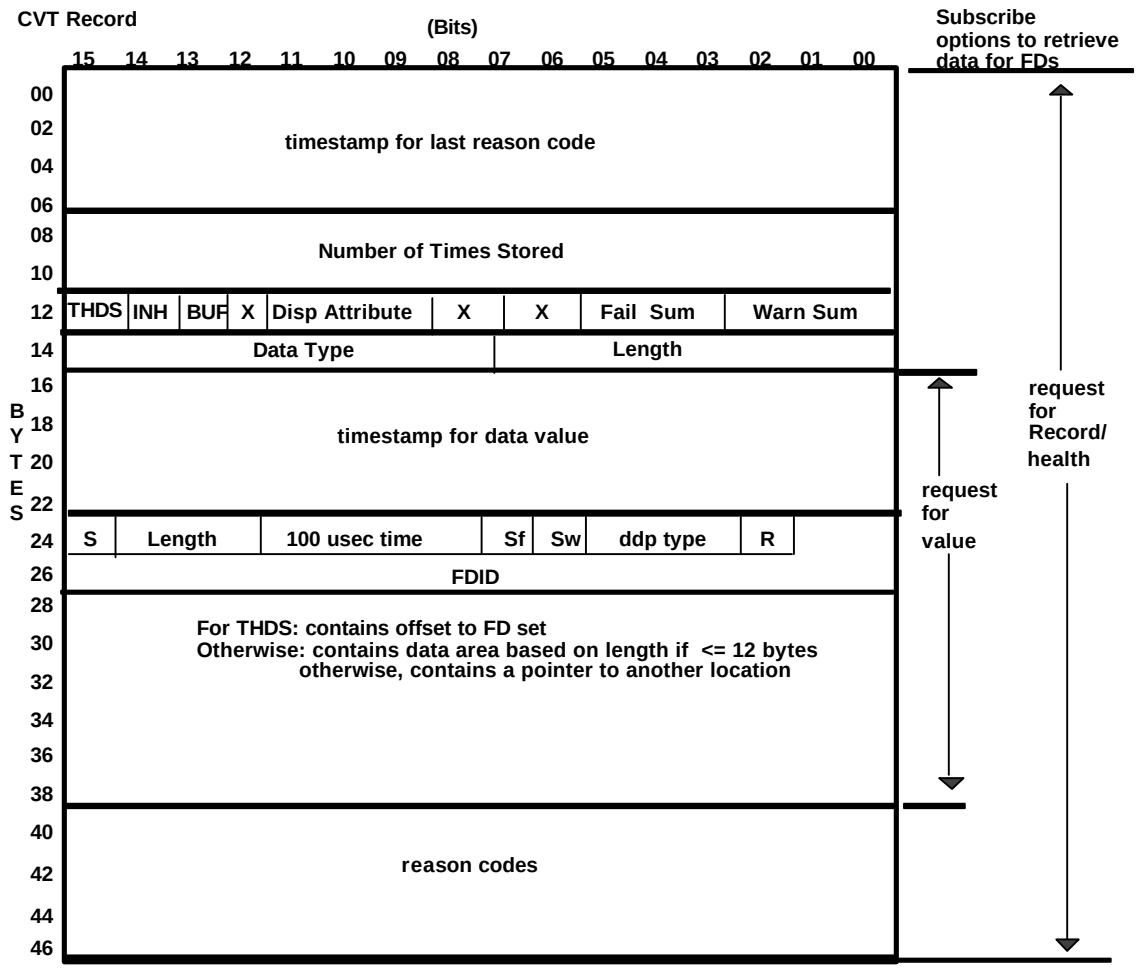
    Notifies the ddp_server to unsubscribe to all FDs

1.3.3 Data Distribution Internal Interfaces

1.3.3.1 Current Value Table Header Format

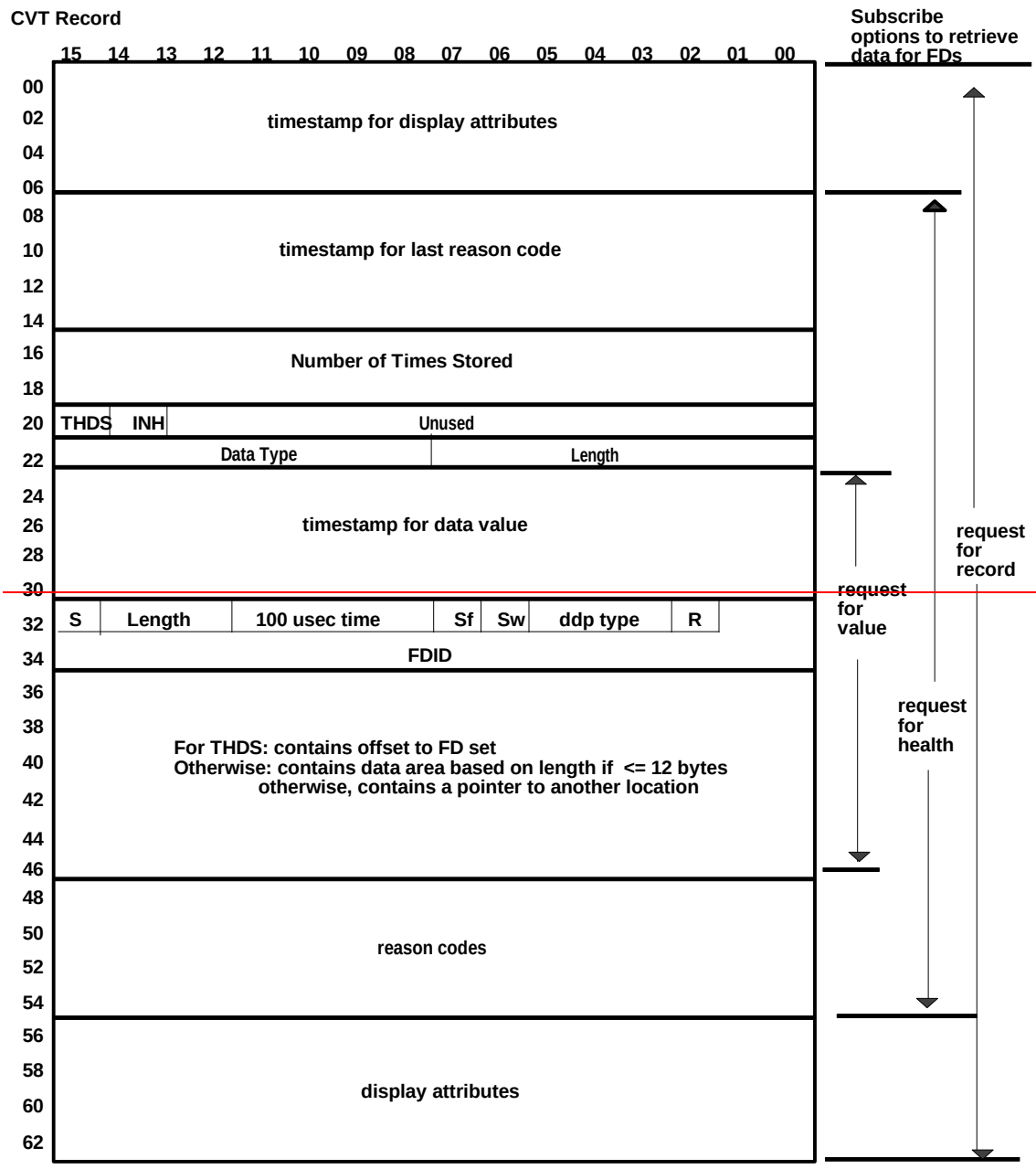


1.3.3.2 Current Value Table Entry Format



The Current Value Table (CVT) contains the latest value, health, and display attributes for each valid FD. The CVT uses shared memory to enable a restrictive set of clients (i.e. performance monitoring, debuggers), to view the contents without impacting the performance of the main processing.

The CVT is an array indexed by FD identifier (FDID). The CVT is initialized with values of certain fields from the OLDB and the THDS for fast access by user applications (i.e engineering units). The Payload Packet is stored in its entirety for performance reasons. A timestamp for the last changed value, a timestamp for the last changed reason code, error/status code, and display attributes are also stored in the CVT.

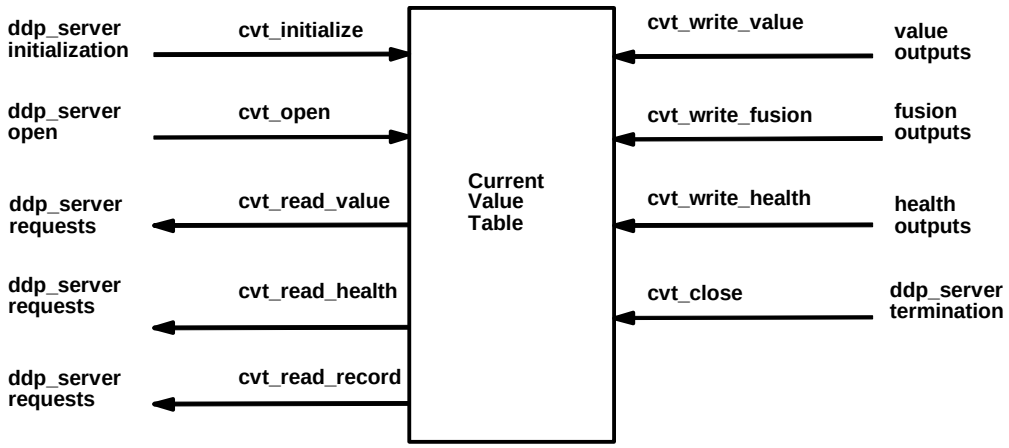


The INH flag indicates the FD is inhibited/enabled. If the FD is inhibited, the CVT record will not be updated when a change arrives. The THDS flag gets set when the CVT is initialized. If set, the FD is a "THDS" set FD.

During CVT initialization, the health bit for each FD is set to "warning" and the reason code is set to "01" indicating the FD data value is uninitialized.

1.3.3.3 CVT Interfaces

The following CVT internal interfaces are provided for maintaining the CVT. They are defined as follows:



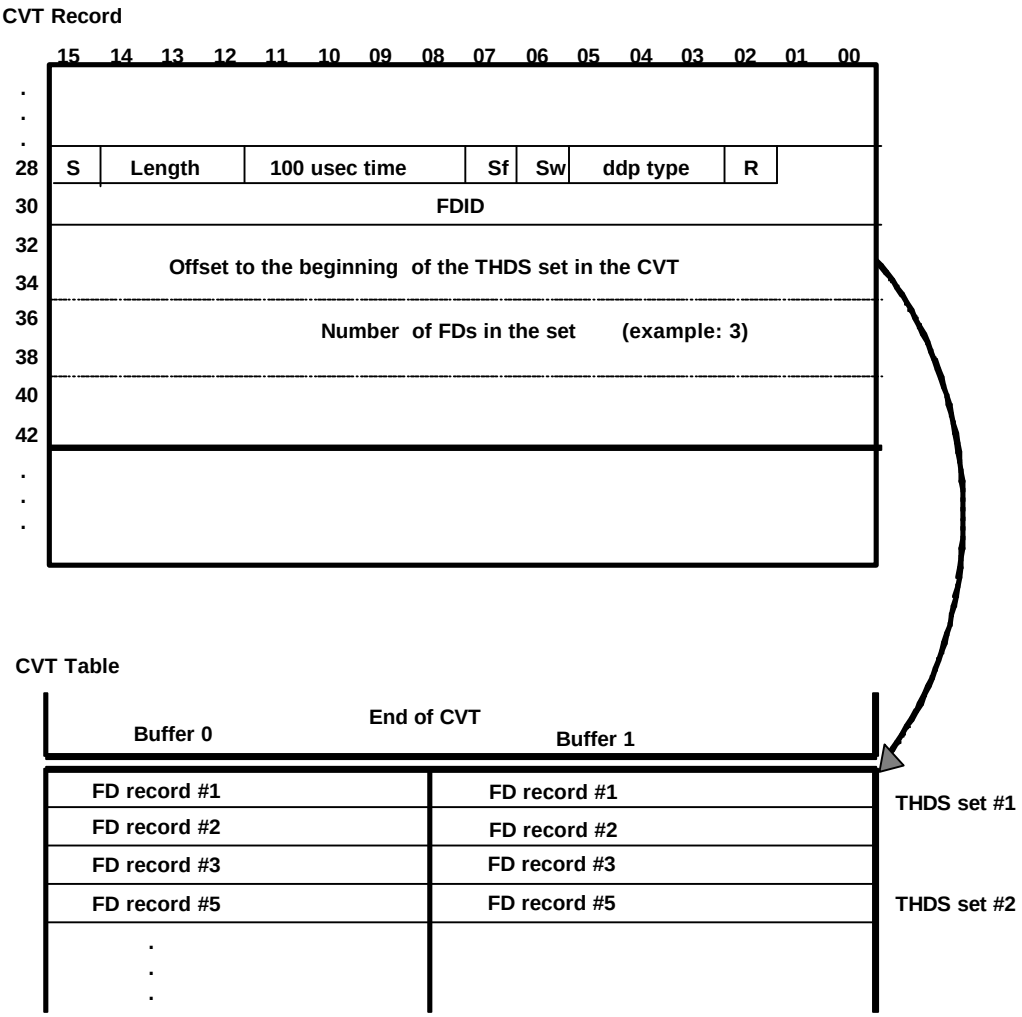
1.3.3.4 CVT THDS Interfaces

CVT initialization reads the THDS file, which contains a list of all THDS FDs, and~~-~~ extends the~~allocates a~~ shared memory segment at the end of the CVT. Whenever a THDS set FD is received, all ~~-~~the FDs in the set are copied to this area in the CVT.

The update count is incremented for the THDS set FD and after all the FDs belonging to the set are copied to the THDS area, their update counts are changed to match the update count of the THDS set FD.

FD services, when using the read CVT api to read directly~~ory~~ from the CVT, must check the record returned to see if it is a THDS set FD. If so, then it will use the offset value in the data section of the record to access the THDS buffer area and proceed to read all the FD records in the data set. FD services must also check the update count of each FD read, making sure it matches the THDS set FD update count to ensure the set data was not in the process of being written while being accessed.

Clients that have subscribed to a THDS set FD will receive a THDS set FD followed by every FD in the THDS set. The update counts are not important to the subscribing client, since the ~~-~~queuing and storing are done by the same process.~~data is queued by the Server.~~



Example:
THDS Set ID = FD4
FD Set FDs = FD1, FD2, FD3

Storing THDS Data:

1. ~~1.~~ FD1, FD2, FD3 received from the gateway. Stored as normal FDs in the CVT.
2. FD4 received from the gateway. The offset and number of FDs are used to copy the CVT records for FD1, FD2, and FD3 into the THDS area.
   ~~Stored as normal FDs in the CVT.~~
3. The records are copied to the THDS for the appropriate buffer.~~2. FD4 received from the gateway. The offset and number of FDs are used to copy the CVT records for FD1, FD2, and FD3 into the THDS area~~
4. The write buffer will be toggled.

~~The records are copied to the THDS for the appropriate buffer.~~

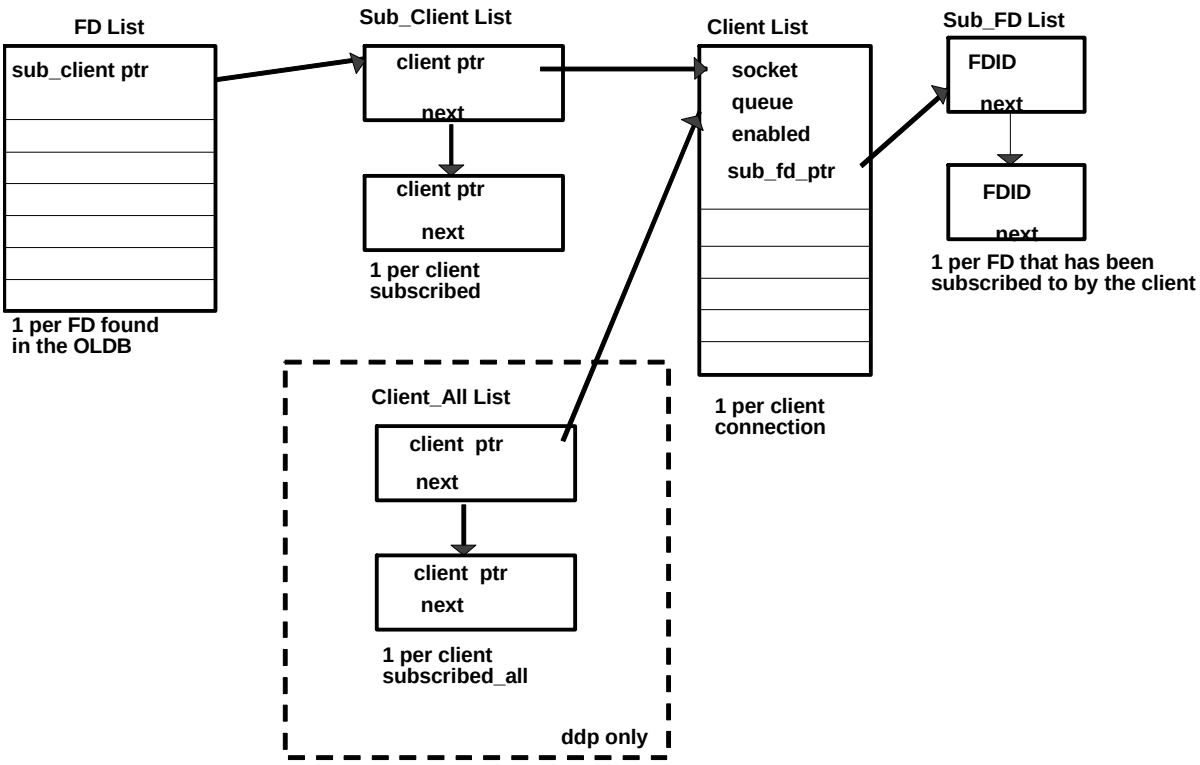
~~1. The write buffer will be toggled.~~

Retrieving THDS Data:

1. To retrieve polled data (CVT read), call cvt_read_record for each of the FDS (FD1, FD2, FD3, and FD4) ~~and~~ for the read buffer area.~~if the update counts match, it will indicate it is a complete set.~~

2. To retrieve queued data, subscribe to FD4. When FD4 is received by the server, FD4, FD1, FD2, and FD3 will be placed on the application queue.

1.3.3.5 Client List Table Formats



Clients on the DDP are the ddp_sender
Clients on the CCWS are the display applications, viewers, user applications, and ddp_collect
Clients on the CCP are the end item managers and ddp_collect

When clients connect to the ddp_server, an enty is added to the CLIENT LIST. The entry contains socket information for notifying the client if values change.

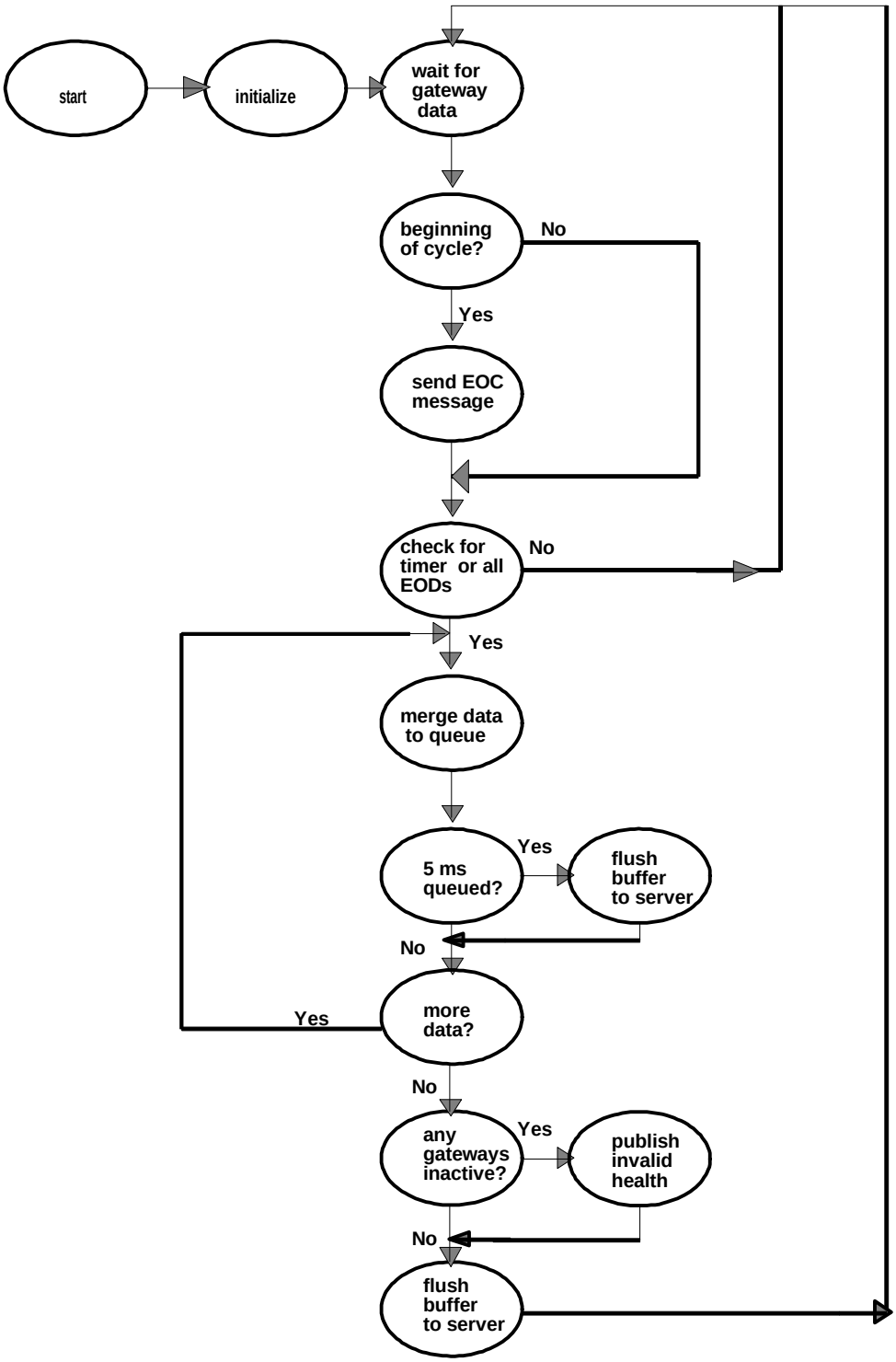
When clients subscribe to FDs, the SUB_FD list is defined with one entry for every request.

The FD_LIST is initialized containing an entry for every FD. It contains a pointer to a linked list for all clients subscribed to that FD. When an FD is published, the SUB Client LIST is searched and an event is placed on the queue for every subscribed client.

Each time an FD changes, the CLIENT_ALL list is searched and an event is placed on the queue for every client

1.3.4 Data Distribution Structure Diagram

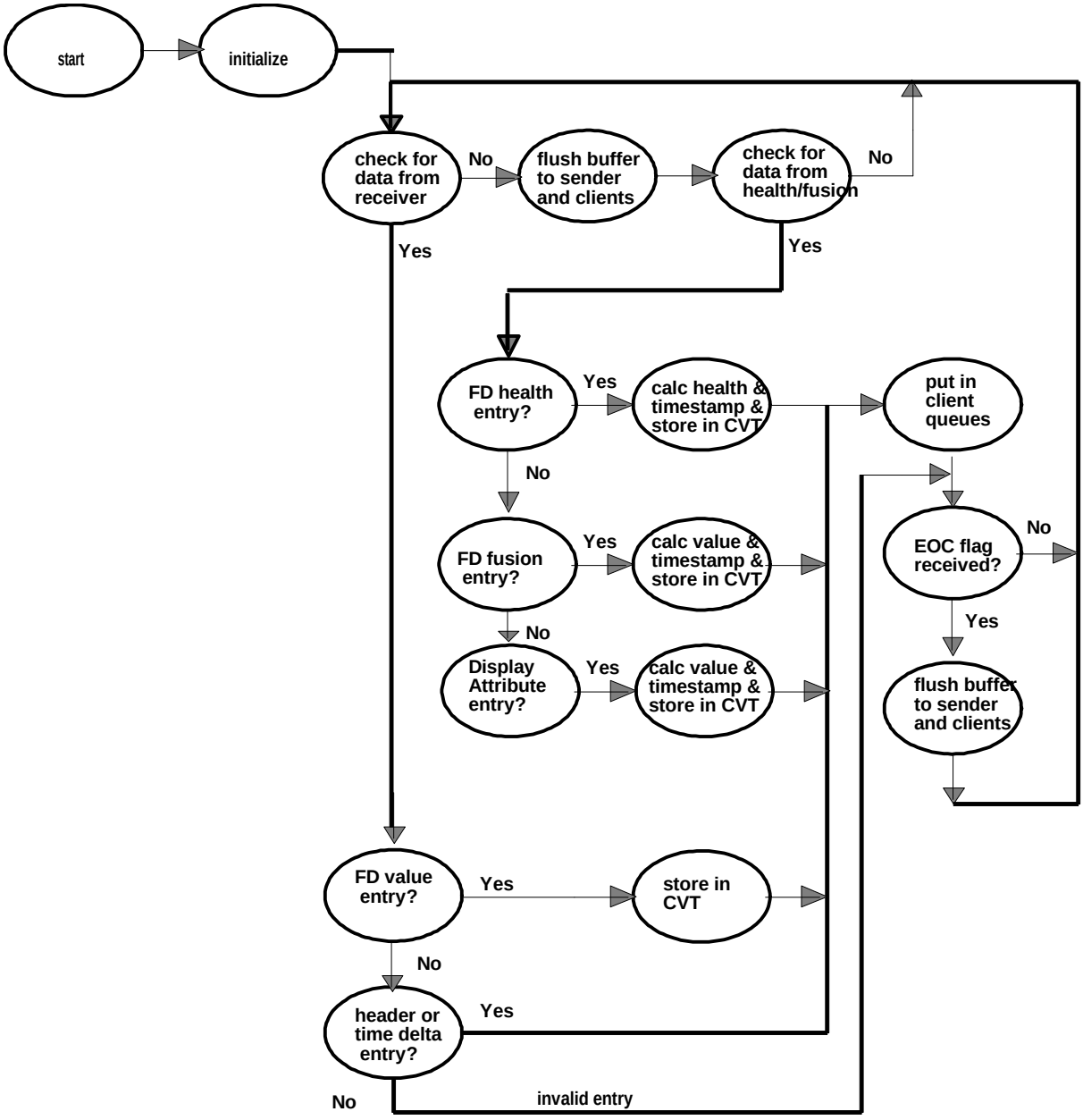
1.3.4.1 Receiver Object



1. **The 1st Header (1every 10ms) is passed to the to ddp_server.**
2. **Time delta packets (max of 9 every 10ms) are passed to the ddp_server as needed.**
3. **An EOC message is passed to the ddp_server at end of 10ms.**

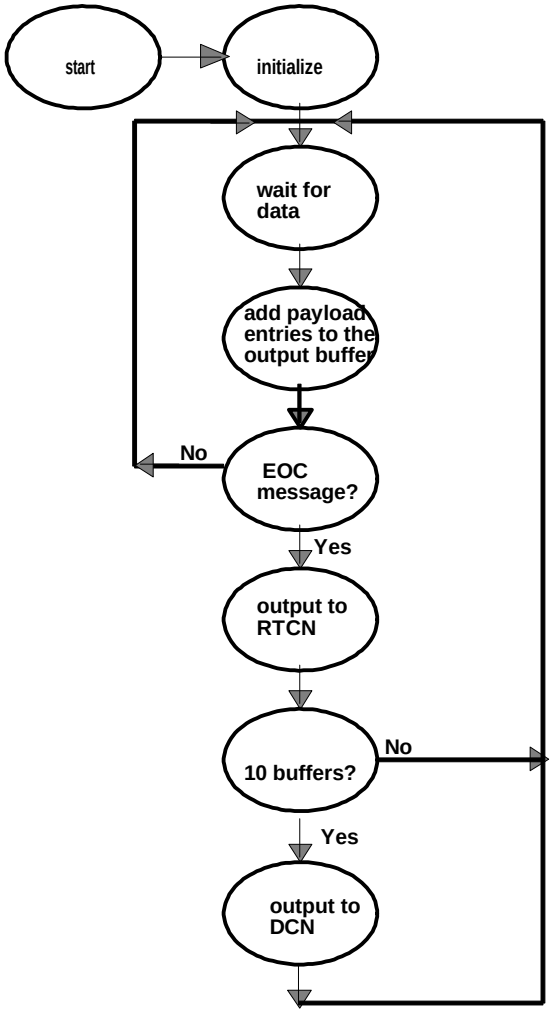
4. **An EOC message is passed if no data is available in 10ms time period.**
5. **The buffer queue is flushed for every 5ms worth of data.**

1.3.4.1.1 Server Object



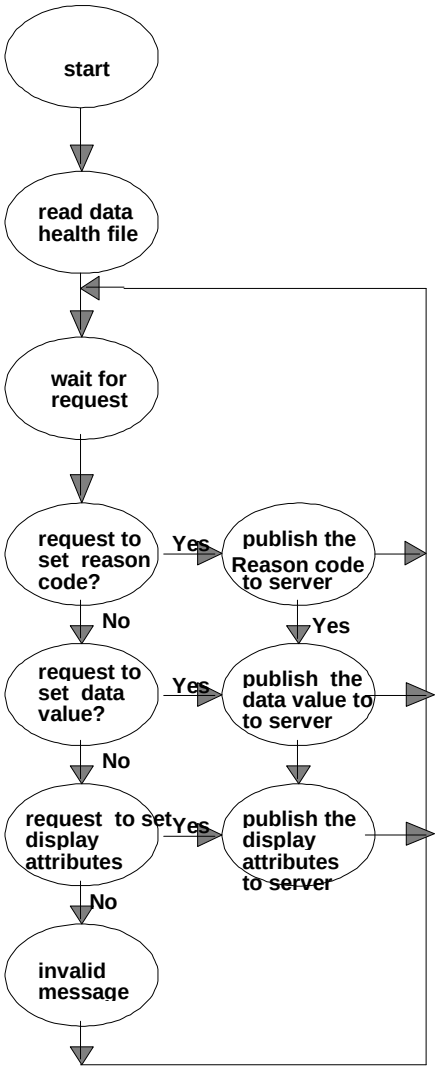
1. **The server process receives data from the receiver process, fusion/health CSCs.**
2. **The server will only process data from the fusion and health CSCs if there is no data to be processed from the ddp_receiver**
3. **The packet being processed will be timestamped (if from fusion or health), and stored in the CVT, unless it is a header, time delta, or EOC message.**
4. **The packet will be placed in the ddp_sender and fusion/health CSCs queues.**
5. **Once the EOC message is processed, the EOC message is placed on the output queue and the queues are flushed.**

1.3.4.1.2 Sender Object



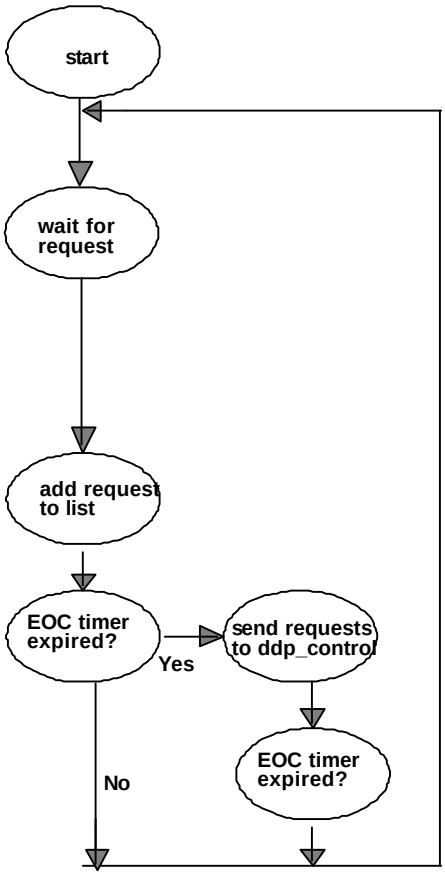
1. **The sender process will receive packets from the ddp_server.**
2. **When the EOC message is received, the sender will output the buffer to the RTCN lan using reliable messaging.**
3. **If 10 packets have been sent to the RTCN, the sender will output the composite buffer to the DCN lan using reliable messaging.**

1.3.4.1.3 Control Object



1. **The control~~mmand~~ process receives commands via IPC Services from the CCP.~~/CCWS.~~**
2. **The control~~mmand~~ process publishes the changed data to the ddp_sever on the DDP.**

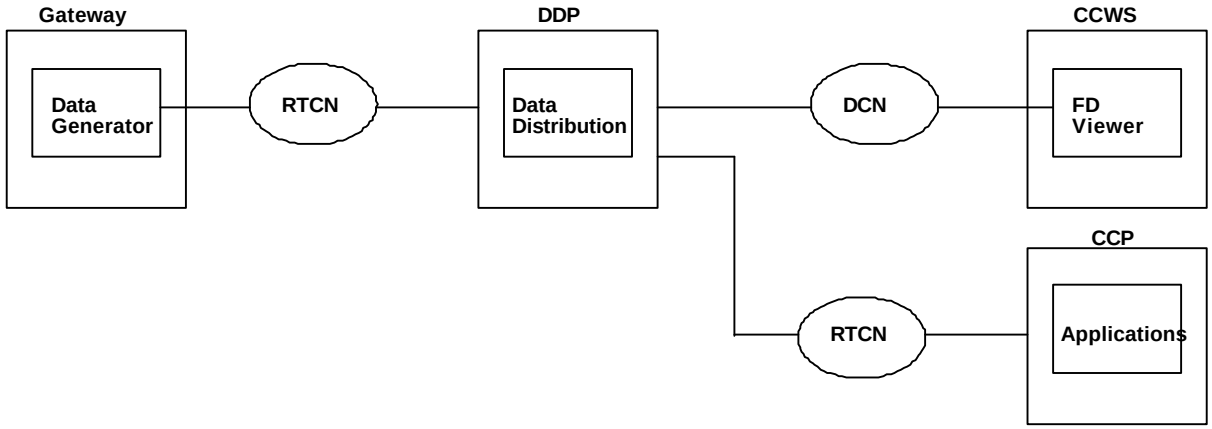
1.3.4.1.4 Collect Object



1. **The collect process receives commands from applications residing on the CCP and sends them to the ddp_control process running on DDP at SSR.**

1.3.5 Data Distribution CIT Test Plan

1.3.5.1 Environment (IDE)



1.3.5.2 Test Tools

1. **Data Generator - used to output up to 20 data streams to the DDP**
2. **PC Goal Like Display - used to compare data values with values on the CCP/CCWS**
3. **ddp_generator - used to construct and feed data streams to DDP on the same platform**
4. **ddp_util - provides a GUI to publish and request data values cyclically or queued and to retrieve packet information from the LAN messages**
5. **cvt_show - display the cvt content**
6. **cvt_stat - display statistical information**

1.3.5.3 Test Cases

Receiver

1. **Verify ddp_receive can establish a connection with ddp_server.**
2. **Verify ddp_receive can establish a connection with LAN.**
3. **Verify ddp_receive can receive multiple data streams from the lan.**
4. **Verify ddp_receive can receive packets at the SSR.**
5. **Verify ddp_receive can time-order FD change packets.**
6. **Verify ddp_receive can publish to ddp_server.**
7. **Verify ddp_receive can realize when a gateway becomes inactive/active.**
8. **verify ddp_reveive sets associated FDs to 'fail' if a gateway becomes inactive.**
9. **verify ddp_receive sets status for incoming FDs, based on Gateway defined for the FDs, when the gateway becomes active again.**

Server

1. **Verify the ddp_server can accept client connections.**
2. **Verify the ddp_server can send FD data to a polling client.**
3. **Verify the ddp_server can send FD data to a client after every change.**
4. **Verify health bits are maintained if FDs are changed from the receiver.**
5. **Verify timestamp information is added to fusion FDs.**
6. **Verify timestamp information is added to FDs undergoing a health change.**
7. **Verify an EOC message is passed through the ddp_server to all clients.**
8. **Verify the ddp_server initililzes correctly.**
   -Read TCID (OLDB) file
   -Read THDS file

Sender

1. **Verify that ddp_sender can establish connection with ddp_server.**
2. **Verify that ddp_sender can establish connection with RTCN and DCN lan.**
3. **Verify that ddp_sender can build 10 ms buffer to the RTCN lan successfully.**
4. **Verify that ddp_sender can build 100 ms buffer to the DCN lan successfully.**

Control

1. **Verify the ddp_control can register with the ddp_server.**
2. **Verify the ddp_control can register with IPC services.**
3. **Verify the ddp_control can accept requests from ddp_collect.**

Collect

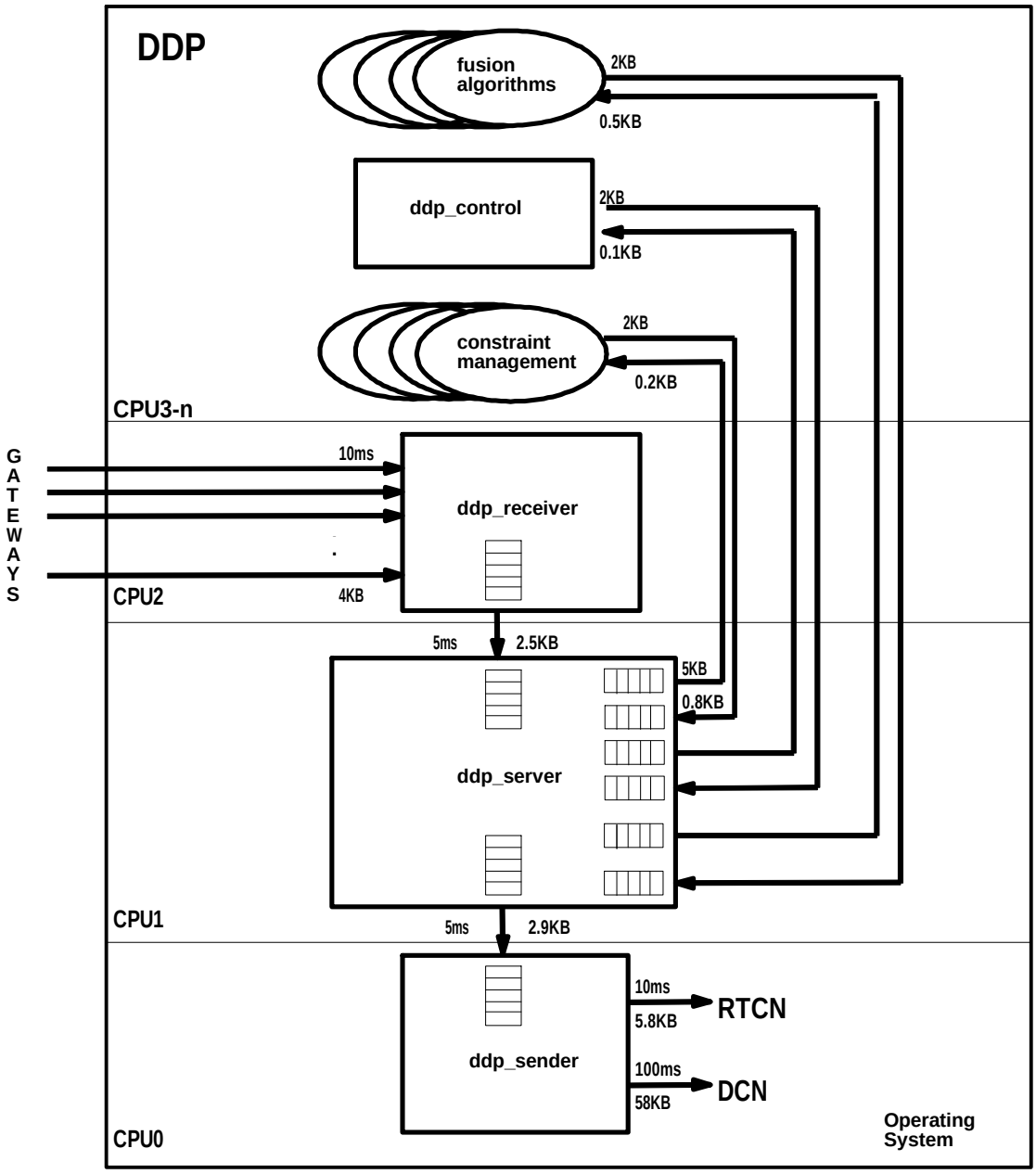
1. **Verify the ddp_collect can receiver requests from applications.**
2. **Verify the ddp_collect can register with IPC services.**
3. **Verify the ddp_collect can send requests to ddp_control.**

APIs

DD API testing will be accomplished by the development of simple code sequences that will exercise all the API calls for initialization/startup, operational, and termination (listed in 1.3.2.7.1).

Appendix A

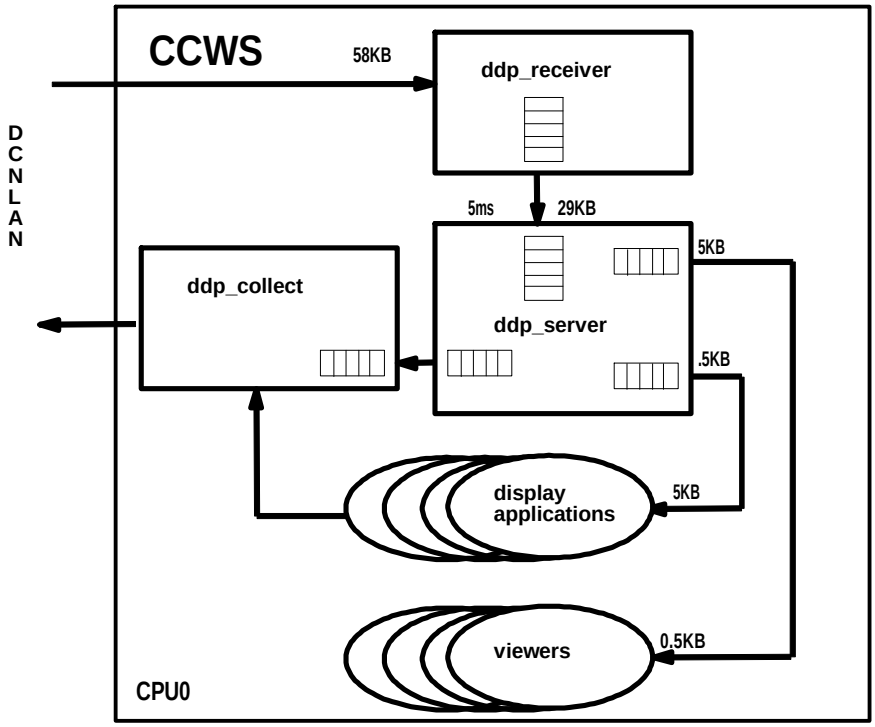
1.3.5.4 Data Distribution Proposed Performance Layout



Data Flow Parameters

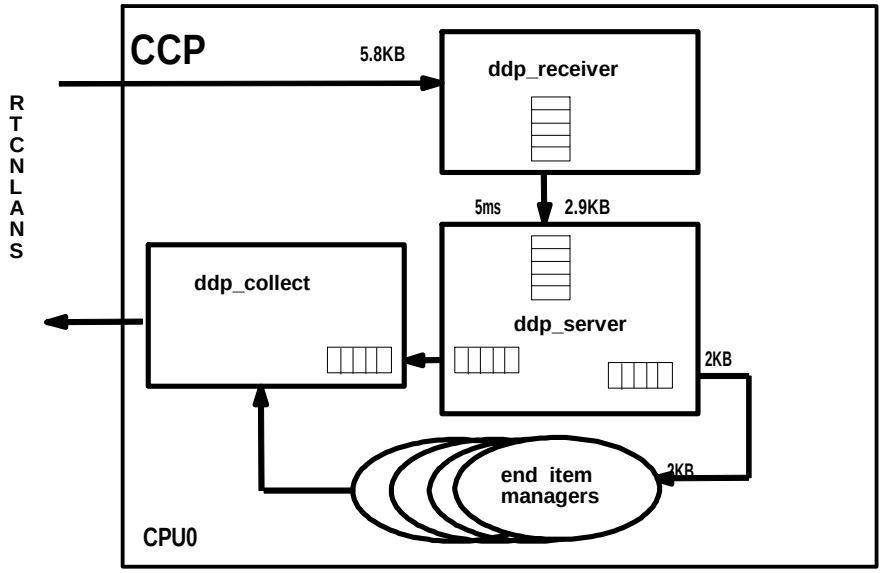
1. **Approximately 40K FDs in the OLDB (FDs from gateway only; 80K FDs with fusion; ~100K FDs with pseudo FDs)**
2. **Average of 8 bytes per packet (32 bit analog)**
3. **4KB bytes per 10ms from the Gateway (8bytes * 500 samples = 4K per ms)**

4. **Approximately 5KB transferred from the receiver every 10ms (4K + hdr/time delta)**
5. **Approximately 5.8KB transferred from the DDP every 10ms (Gateway data and fusion/health data)**
6. **Approximately 20ms latency (10ms in the receiver collecting Gateway data, and 10ms in the server/sender)**



Data Flow Parameters

1. **Approximately 58K will be received every 100ms from the DDP**
2. **Approximately 5KB transferred from the server to the clients every 100ms**



Data Flow Parameters

1. **Approximately 5.8K will be received every 10ms from the DDP**
2. **Approximately 2KB transferred from the server to the clients every 100ms**

Queue Private Library Object

Queues are used to buffer communication between the ddp_server and client applications. Queues are defined to be allocated memory where changed packets are stored in sequence to be sent to requesting processes. The queues minimize the number of system calls (which are expensive), and allow for the elasticity between the processes. The flush rate for the processes varies, based on the latency.....

• ddp_queue_create	create a queue
• ddp_queue_clear	clear a queue contents
• ddp_queue_destroy	destroy a queue
• ddp_queue_get_bufsize	retrieve queue allocated size
• ddp_queue_get_size	retrieve # bytes in queue
• ddp_queue_get_free_space	retrieve # available bytes in queue
• ddp_get_enqueue_buf	get write buffer from queue
• ddp_queue_alloc_bytes	allocate the specified bytes
• ddp_get_dequeue_buf	get read buffer from queue
• ddp_get_free_bytes	deallocate specified amount
• ddp_queue_peek_at_data	copy data into buffer
• ddp_dequeue_data	read data from queue
• ddp_enqueue_data	copy data into queue
• ddp_queue_resize	rsize queue while preserving contents
• ddp_queue_cache_grow	grow the queue cache to add fd
• ddp_queue_reset_fd	reset queue for file desc. and I/O dir
• ddp_queue_from_fd	find queue from file desc. and I/O dir
• ddp_queue_cache_flush	flush output queue to their sockets
• ddp_queue_count_cycle_bytes	raise data cycle size by byte count
• ddp_queue_get_cycle_count	get data cycle size counter
• ddp_queue_clear_cycle_count	clear data cycle size counter
• ddp_get_enqueue_buf	get write buffer from queue

Socket Private Library Object

The socket library contains socket management utilities. The TCP socket routines are responsible for the server and clients communications.

• ddp_accept_socket	accept client connection
• ddp_close_socket	close client connection
• ddp_set_connect_timeout	enable/disable socket connect timeout
• ddp_create_socket	initiate connection
• ddp_open_socket	connect client to server
• ddp_shutdown_socket_no_send	close outgoing halp of a socket
• ddp_write_from_queue	write into socket from queue
• ddp_read_from_queue	read into queue from socket
• ddp_send_accept	send accept packet
• ddp_send_connect	send connect packet
• ddp_send_control	send control packet
• ddp_send_cycle	send cycle packet
• ddp_send_disconnect	send disconnect packet
• ddp_send_enable	send enable packet
• ddp_send_publish	send publish packet
• ddp_send_response	send response packet
• ddp_send_start	send start packet
• ddp_send_stop	send stop packet
• ddp_send_subscribe	send subscribe packet
• ddp_send_unsubscribe	send unsubscribe packet
• ddp_send_unpublish	send unpublish packet
• ddp_send_value	send value packet
• ddp_alloc_circular_buffer	allocate a circular buffer
• ddp_write_buffer	write the buffer out

APPENDIX C

Statement of Work (Thor Delivery)

- Provide performance data for system modeling.
- Confirm and or modify system data flow for FD Data Distribution and provide the appropriate updates to the SDD.
- Provide the capability for the Data Distribution function to be utilized in both Operational and Application configurations.
- Add support for Application Change Data.

DDP Data ~~Merger~~ Function

- Collect Application Change Data packets from all CCPs at the system synchronous rate.
- Collect Application Change Data packets from CCWS on demand.
- Merge Gateway Change Data and Application Change Data in to a single a stream.
- Place requested FDs in queues for the ~~Data Constraint Function~~ Constraint Manager.
- Define and provide a method to send System Default Display Data Attribute Values (dynamic limits and animation).
- Maintain statistics on packet rates and data rates. ~~and CPU utilization.~~

CCP Data Function

- Provide an output queue for user Application Derived FDs and transmits them to the DDP at system synchronous rate.
- Define and provide a method to send System Default FD Display Attribute,
- Place requested FDs in queues for the ~~Data Constraint Function~~ Constraint Manager.

HCI Data Function

- Provide the capability for applications at the HCI to set FD values and/or display attributes.

General

- Maintain statistics on packet rates, data rates and errors ~~and CPU utilization~~.
- Provide performance data for system modeling.
- Confirm and or modify system data flow for FD Data Distribution and provide the appropriate updates to the SDD.
- Provide the capability for the Data Distribution function to be utilized in both Operational and application development ~~Test Bed~~ configuration.
- Provide logging of error, performance, and state chagne information.
- Baseline system messages using the System Message Catalog to include message and help text.
- Support FD override capability.

Performance Requirements from SLS

- The “system maximum data bandwidth” shall support 25,000 end item changes per second continuously.
- The system shall support 50,000 end item measurement changes in a given second without losing any data.
- The system shall support 1,000 end item measurement changes during a 10 millisecond period.
- One user’s test applications shall be able to read 10,000 measurements, and “verify” them in a single second in an unloaded system.
- The data distribution function shall support the “system maximum data bandwidth”, plus 5,000 (20%) Data Fusion updates per second.
- The time from a measurement change being detected by the system until that measurement is available for retrieval shall be less than five seconds. (2.2.2.1.12)

Other Data Distribution Related Requirements from SLS

- The RTPS shall maintain the current value of all Measurement FDs for access by application SW. (2.2.3.1.3)
- The RTPS shall provide the capability for applications to request all changes of a selected set of Measurement FDs and have them provided along with time of change and health at the time of change. (2.2.3.1.4)
- The RTPS shall provide changed measurement data to system and user applications at the System Synchronous Rate. (2.2.3.1.5)
- The RTPS shall provide changed measurement data to display applications at the Display Synchronous Rate.
- CLCS shall provide the user the capability to create Pseudo FDs and generate their values using Test Applications.
- CLCS shall provide the capability for Pseudo FD values to be persistent between different tests.
- CLCS shall provide the user the capability to choose which Pseudo FDs are to be persistent.

High level derived requirements

1. Data Distribution shall provide an application interface allowing an application to assign the following to a FD or a list of FDs:
   - Data value
   - Display attribute
2. Data Distribution shall provide a mechanism to collect the changed Application Derived FDs into an Application Change Data packet at the CCP and send the packet to Data Distribution at the DDP.
3. Data Distribution shall sent the "set FD value/display attribute" request at the HCI upon request.
4. Data Distribution shall receive Application Change Data Packets at the DDP.
5. Data Distribution shall collect Application Change Data packets received from all the CCPs and update the CVT with the changed data.
6. Data Distribution shall include the changed Application Derived FD's in the Data Distribution packets at the DDP and output to the RTCN and DCN.
7. Data Distribution shall support processing of data in a Time Homogenous Data Set.
8. Data Distribution shall notify Subsystem Integrity immediately whenever a failure occurs that causes Data Distribution to cease processing change data.
9. Data Distribution shall provide an application program interface (API) for Subsystem Integrity to obtain the following information:
   - Overall health of Data Distribution
   - Error situations (e.g. Failure to setup Network communication; errors that cause Data Distribution to terminate.)
   - Statistics on packet size and data rate
10. Data Distribution shall incorporate Constraint Notifications in the RTCN Data Distribution packets. (Note: This is a potential requirement to be levied on Data Distribution by Constraint Manager.)
11. Data Distribution shall generate the following test tools:
    a. The capability to construct and feed data streams to Data Distribution residing on the same platform. The number of streams and contents of the data streams will be based on FD names, values and rates defined in an ASCII file.
    b. An interactive tool that allows the user to:
       - Subscribe to a list of FDs and display values on the screen.
       - Publish a list of FDs at specific rate
       - Retrieve the packets from the DCN or RTCN and print the header time, length of packet and number of packets being sent.
    c. Capability to display the CVT content.

APPENDIX D~~C~~

Statement of Work (Redstone Delivery)

- Provide performance data for system modeling.
- Confirm and or modify system data flow for FD Data Distribution.
- Provide the capability for the Data Distribution function to be utilized in both Operational and Application configurations.

DDP Data Merger Function

- Collect Gateway Change Data packets from all gateways at the system synchronous rate.
- Collect Application Change Data packets from all CCPs at System synchronous rate (*No Application Change Data packets until Thor Delivery*).
- Merge Gateway Change Data and Application Change Data in to a single a stream ordered to the nearest 0.1 ms. (*No Application Change Data until Thor Delivery*).
- Merge health data into data element from health table.
- Place requested FDs in queues for the Data Fusion Function.
- Place requested FDs in queues for the Data Health Function.
- Place requested FDs in queues for the Data Constraint Function (*Data Constraint Function is not part of Redstone Delivery*).
- Transmit this data at system synchronous rate on the RTCN.
- Transmit this data at display synchronous rate on the DCN.
- Define and provide a method to send System Default Display Data Attribute Values. (*A placeholder will be reserved in the CVT for default Display Data Attribute values. Setting mechanism will be defined for the Thor delivery*).
- Maintain statistics on packet rates, data rates, and CPU utilization.

CCP Data Function

- Collect RTCN Change Data Packets from the DDP at system synchronous rate.
- Place requested FDs in queues for System and User Application.
- Provide an output queue for user Application Derived FDs and transmits them to the DDP at system synchronous rate. (*No user Application Derived FDs to transmit in Redstone*).
- Maintain statistics on packet rates, data rates, and CPU utilization.

HCI Data Function

- Collect DCN Change Data Packets from the DDP at display synchronous rate.
- Place requested FDs in queues for System and User Application.
- Maintain statistics on packet rates, data rates, and CPU utilization.

Current Value Table Function

- Maintain in the DDPs, CCPs and HCIs a Current Value Table that contains for all FDs the current data value, its health and time of last change.
- Support all FD type including Time Homogenous and Multiword data.
- Provide separation of data for different flow zones. (Added during kickoff meeting)